Temporal 101

GO

Version: 20230910a-replay

Temporal 101

00 About this Workshop

01 What is Temporal?

02 Developing a Workflow

03 Executing a Workflow

04 Viewing Workflow Execution History

05 Modifying an Existing Workflow

06 Developing an Activity

07 Handling Activity Failure

08 Understanding Workflow Execution

09 Conclusion

Logistics

- Introductions
- Schedule
- Facilities
- WiFi
- Course conventions ("workflow" vs. "Workflow")
- Asking questions
- Getting help with exercises

During this course, you will

- Learn the basic architecture of the Temporal platform
- Develop and execute Workflows and Activities using the Go SDK
- Use the Web UI to gain insight into current and previous executions
- Experiment with failures and retries
- Understand how a Temporal cluster orchestrates execution

Temporal 101

00 About this Workshop

01 What is Temporal?

02 Developing a Workflow

03 Executing a Workflow

04 Viewing Workflow Execution History

05 Modifying an Existing Workflow

06 Developing an Activity

07 Handling Activity Failure

08 Understanding Workflow Execution

09 Conclusion

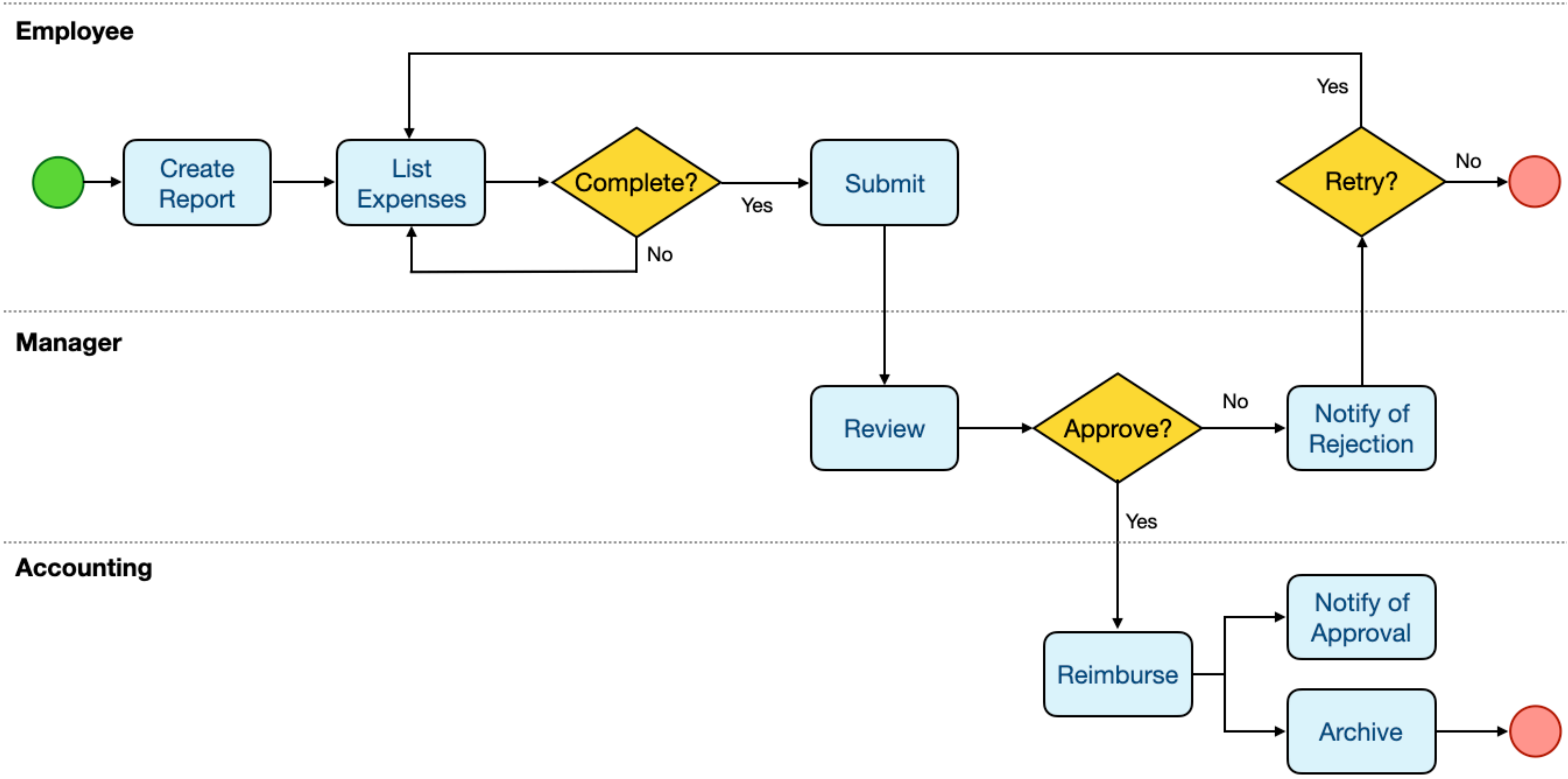
Introducing Temporal

- **The Temporal Platform is a durable execution system for your code**
- **Temporal applications are created using *Workflows***
  - Like other applications, you develop them by writing code
    - The code you write is the code that is executed at runtime
  - Unlike other applications, Temporal Workflows are resilient
    - They can run for years, surviving both server and application crashes

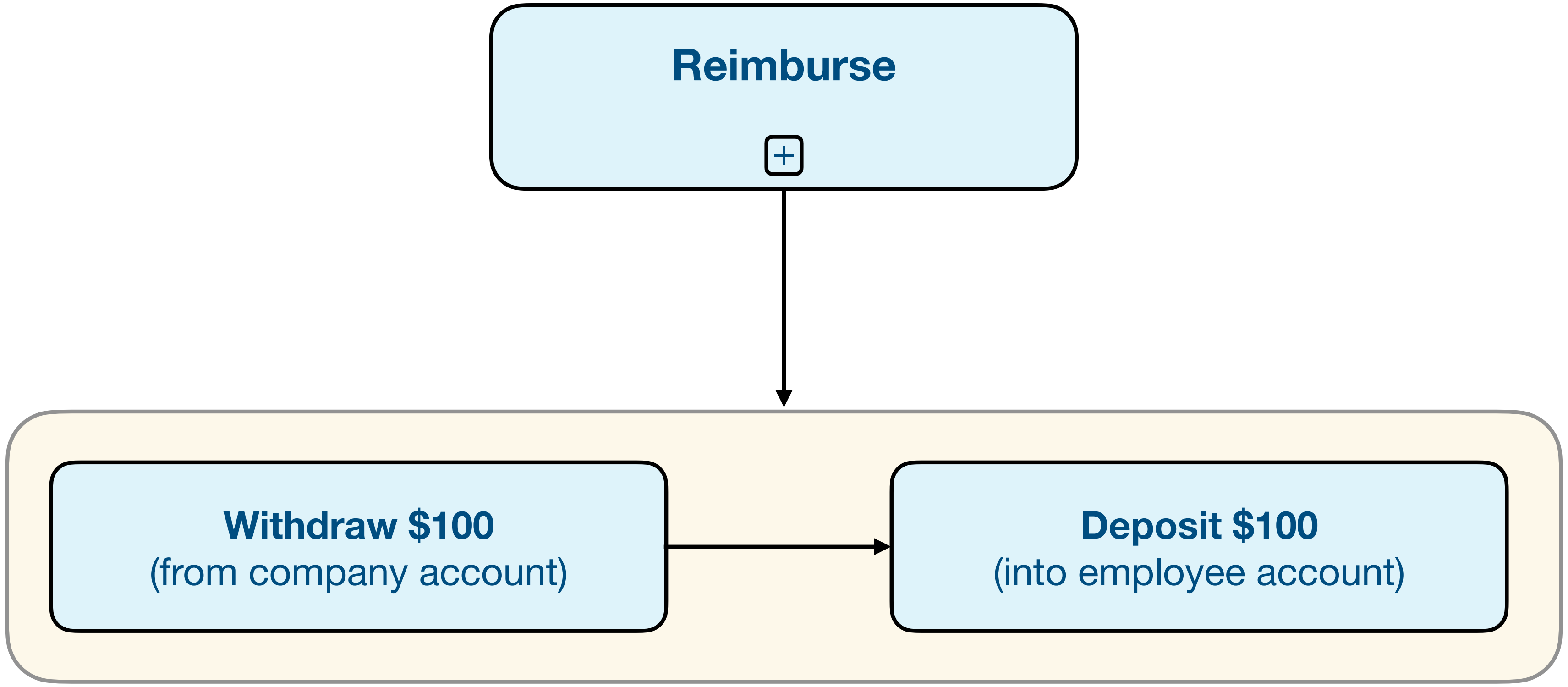
What is a Workflow?

- **Conceptually, a workflow is a sequence of steps**
- **You probably have experience with workflows from everyday life**
  - Using a mobile app to transfer money
  - Buying concert tickets
  - Booking a vacation
  - Ordering a pizza
  - Filing an expense report

Workflow Example: Expense Report

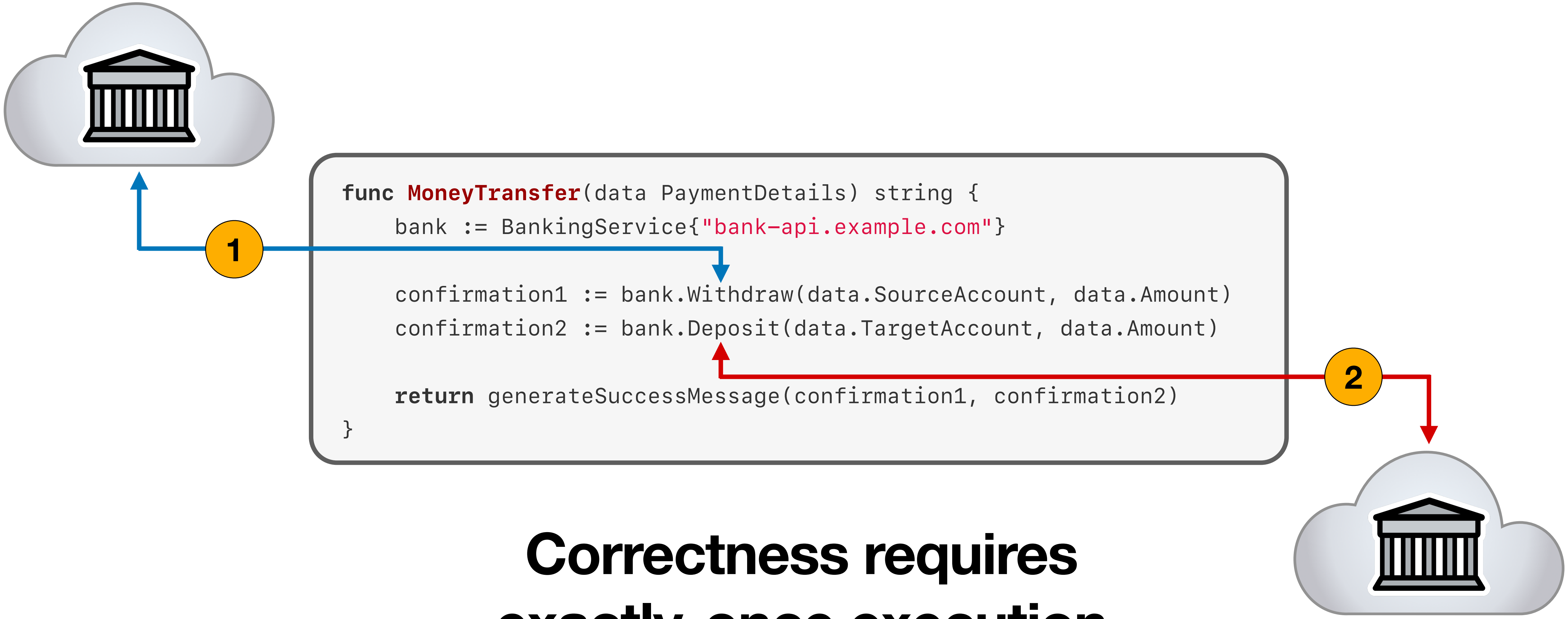


Workflow Example: Reimbursement



Correctness requires exactly-once execution

This Workflow Is a Distributed System

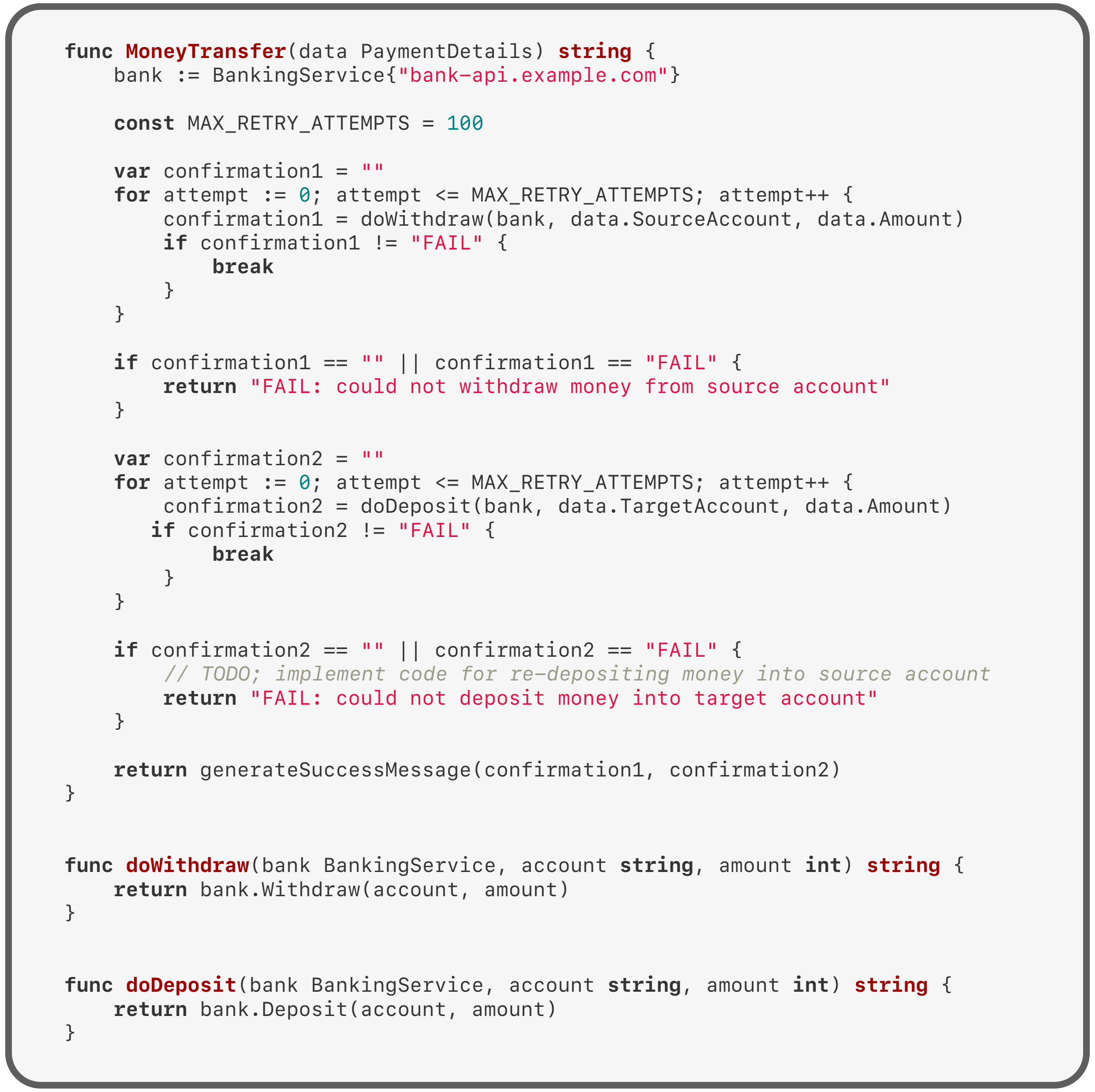


```
func MoneyTransfer(data PaymentDetails) string {
    bank := BankingService{"bank-api.example.com"}

    confirmation1 := bank.Withdraw(data.SourceAccount, data.Amount)
    confirmation2 := bank.Deposit(data.TargetAccount, data.Amount)

    return generateSuccessMessage(confirmation1, confirmation2)
}
```

Correctness requires exactly-once execution

Failure Mitigation: Retries

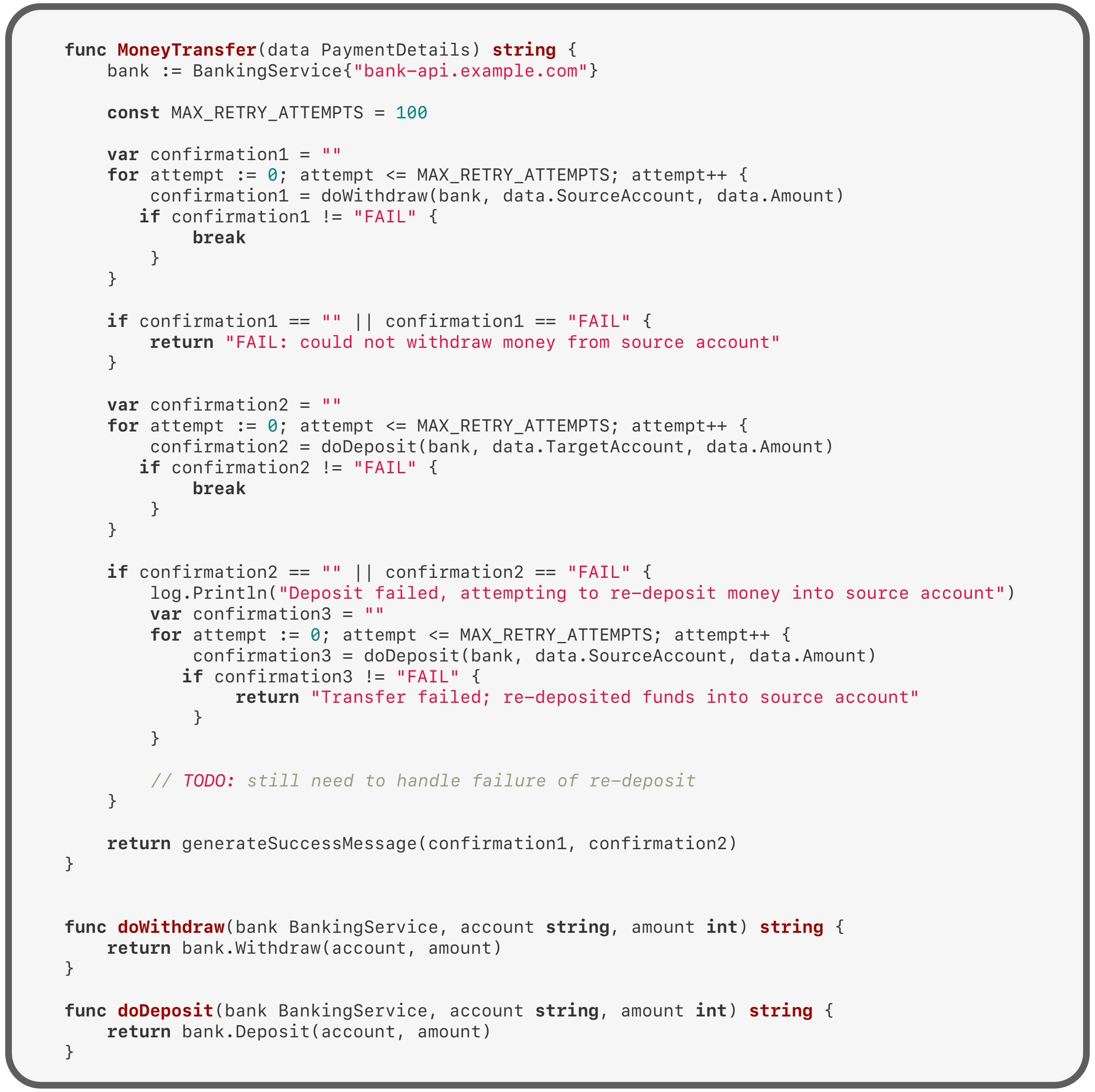
The same code, after adding support for retries



```
func MoneyTransfer(data PaymentDetails) string {
    bank := BankingService{"bank-api.example.com"}

    const MAX_RETRY_ATTEMPTS = 100

    var confirmation1 = ""
    for attempt := 0; attempt <= MAX_RETRY_ATTEMPTS; attempt++ {
        confirmation1 = doWithdraw(bank, data.SourceAccount, data.Amount)
        if confirmation1 != "FAIL" {
            break
        }
    }

    if confirmation1 == "" || confirmation1 == "FAIL" {
        return "FAIL: could not withdraw money from source account"
    }

    var confirmation2 = ""
    for attempt := 0; attempt <= MAX_RETRY_ATTEMPTS; attempt++ {
        confirmation2 = doDeposit(bank, data.TargetAccount, data.Amount)
       if confirmation2 != "FAIL" {
            break
        }
    }

    if confirmation2 == "" || confirmation2 == "FAIL" {
        // TODO; implement code for re-depositing money into source account
        return "FAIL: could not deposit money into target account"
    }

    return generateSuccessMessage(confirmation1, confirmation2)
}


func doWithdraw(bank BankingService, account string, amount int) string {
    return bank.Withdraw(account, amount)
}


func doDeposit(bank BankingService, account string, amount int) string {
    return bank.Deposit(account, amount)
}
```

Failure Mitigation: Compensations

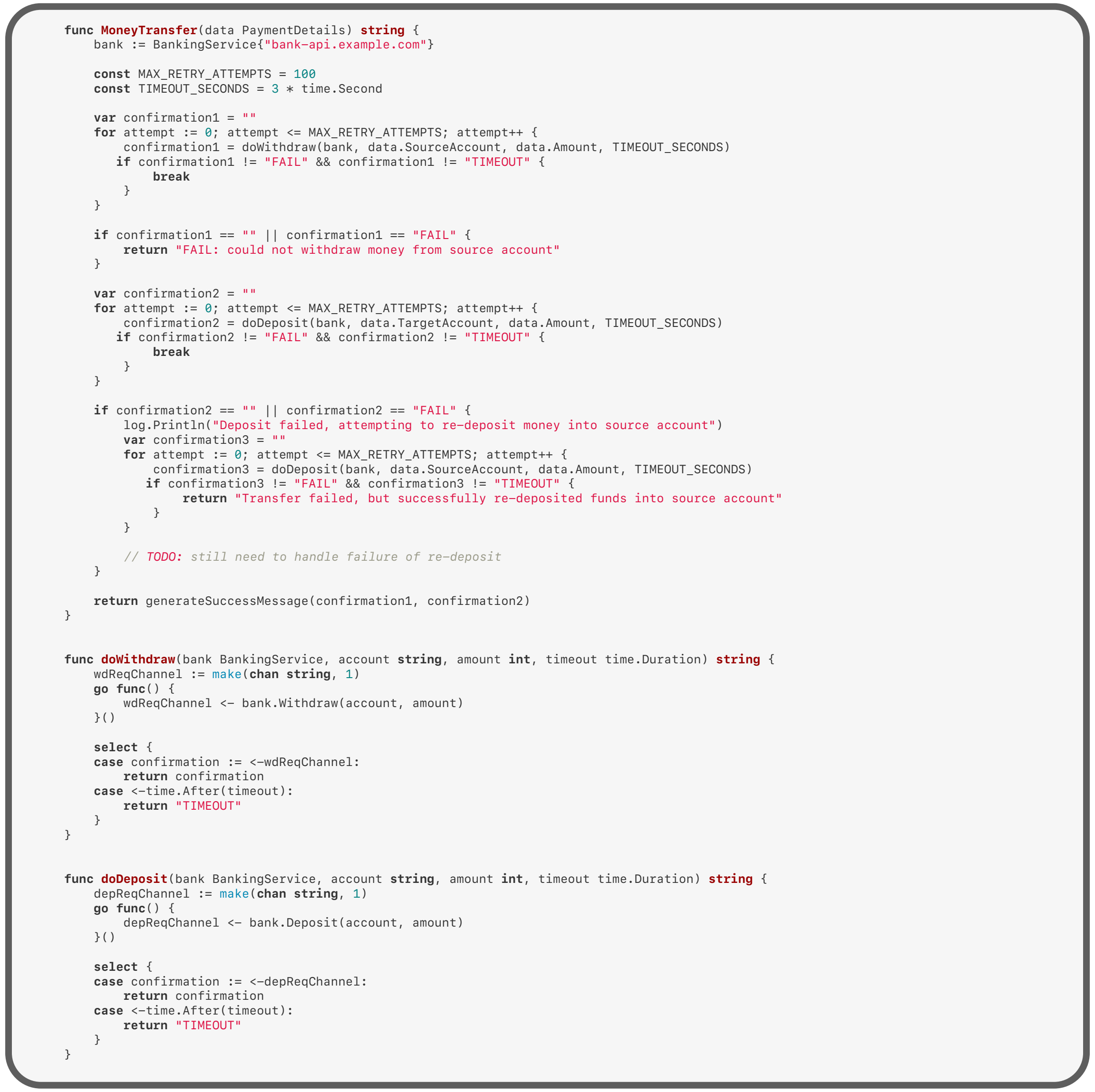
After adding code to recover from a failed deposit



```go
func MoneyTransfer(data PaymentDetails) string {
    bank := BankingService{"bank-api.example.com"}

    const MAX_RETRY_ATTEMPTS = 100

    var confirmation1 = ""
    for attempt := 0; attempt <= MAX_RETRY_ATTEMPTS; attempt++ {
        confirmation1 = doWithdraw(bank, data.SourceAccount, data.Amount)
       if confirmation1 != "FAIL" {
            break
        }
    }

    if confirmation1 == "" || confirmation1 == "FAIL" {
        return "FAIL: could not withdraw money from source account"
    }

    var confirmation2 = ""
    for attempt := 0; attempt <= MAX_RETRY_ATTEMPTS; attempt++ {
        confirmation2 = doDeposit(bank, data.TargetAccount, data.Amount)
       if confirmation2 != "FAIL" {
            break
        }
    }

    if confirmation2 == "" || confirmation2 == "FAIL" {
        log.Println("Deposit failed, attempting to re-deposit money into source account")
        var confirmation3 = ""
        for attempt := 0; attempt <= MAX_RETRY_ATTEMPTS; attempt++ {
            confirmation3 = doDeposit(bank, data.SourceAccount, data.Amount)
           if confirmation3 != "FAIL" {
                return "Transfer failed; re-deposited funds into source account"
            }
        }

        // TODO: still need to handle failure of re-deposit
    }

    return generateSuccessMessage(confirmation1, confirmation2)
}


func doWithdraw(bank BankingService, account string, amount int) string {
    return bank.Withdraw(account, amount)
}

func doDeposit(bank BankingService, account string, amount int) string {
    return bank.Deposit(account, amount)
}
```

Failure Mitigation: Timeouts

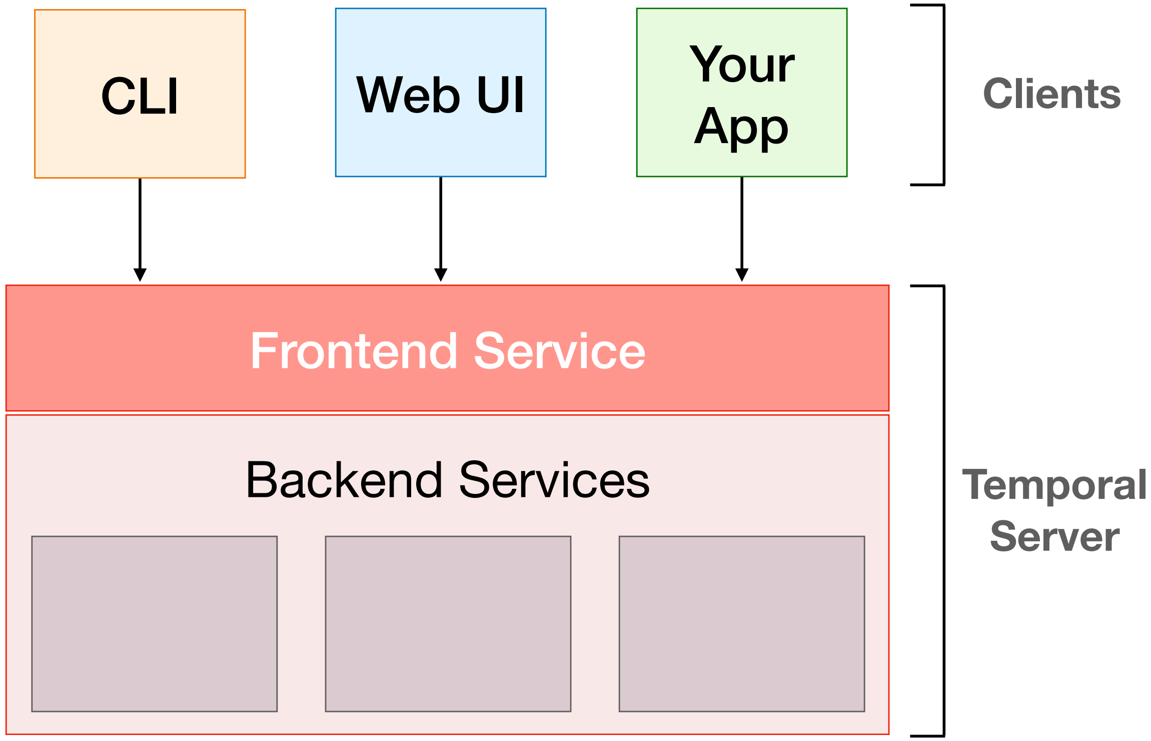
After adding support for request timeouts



```go
func MoneyTransfer(data PaymentDetails) string {
    bank := BankingService{"bank-api.example.com"}

    const MAX_RETRY_ATTEMPTS = 100
    const TIMEOUT_SECONDS = 3 * time.Second

    var confirmation1 = ""
    for attempt := 0; attempt <= MAX_RETRY_ATTEMPTS; attempt++ {
        confirmation1 = doWithdraw(bank, data.SourceAccount, data.Amount, TIMEOUT_SECONDS)
       if confirmation1 != "FAIL" && confirmation1 != "TIMEOUT" {
            break
        }
    }

    if confirmation1 == "" || confirmation1 == "FAIL" {
        return "FAIL: could not withdraw money from source account"
    }

    var confirmation2 = ""
    for attempt := 0; attempt <= MAX_RETRY_ATTEMPTS; attempt++ {
        confirmation2 = doDeposit(bank, data.TargetAccount, data.Amount, TIMEOUT_SECONDS)
       if confirmation2 != "FAIL" && confirmation2 != "TIMEOUT" {
            break
        }
    }

    if confirmation2 == "" || confirmation2 == "FAIL" {
        log.Println("Deposit failed, attempting to re-deposit money into source account")
        var confirmation3 = ""
        for attempt := 0; attempt <= MAX_RETRY_ATTEMPTS; attempt++ {
            confirmation3 = doDeposit(bank, data.SourceAccount, data.Amount, TIMEOUT_SECONDS)
           if confirmation3 != "FAIL" && confirmation3 != "TIMEOUT" {
                return "Transfer failed, but successfully re-deposited funds into source account"
            }
        }

        // TODO: still need to handle failure of re-deposit
    }

    return generateSuccessMessage(confirmation1, confirmation2)
}


func doWithdraw(bank BankingService, account string, amount int, timeout time.Duration) string {
    wdReqChannel := make(chan string, 1)
    go func() {
        wdReqChannel <- bank.Withdraw(account, amount)
    }()

    select {
    case confirmation := <-wdReqChannel:
        return confirmation
    case <-time.After(timeout):
        return "TIMEOUT"
    }
}


func doDeposit(bank BankingService, account string, amount int, timeout time.Duration) string {
    depReqChannel := make(chan string, 1)
    go func() {
        depReqChannel <- bank.Deposit(account, amount)
    }()

    select {
    case confirmation := <-depReqChannel:
        return confirmation
    case <-time.After(timeout):
        return "TIMEOUT"
    }
}
```

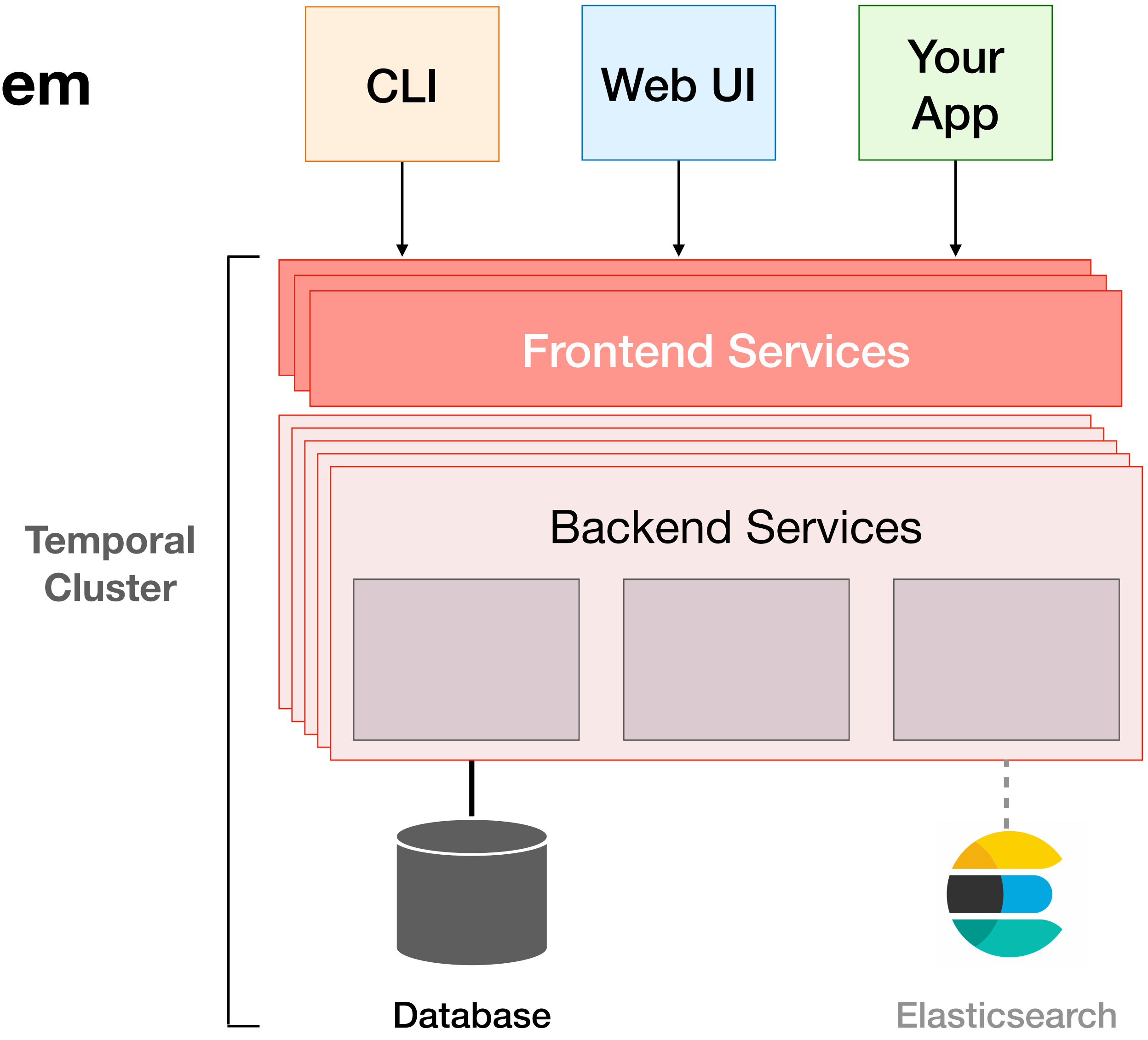
Architectural Overview: Temporal Server

- **Consists of multiple services**
  - Each service is horizontally scalable
  - The frontend service is an API gateway
  - Clients are external to the server and interact only with the frontend service



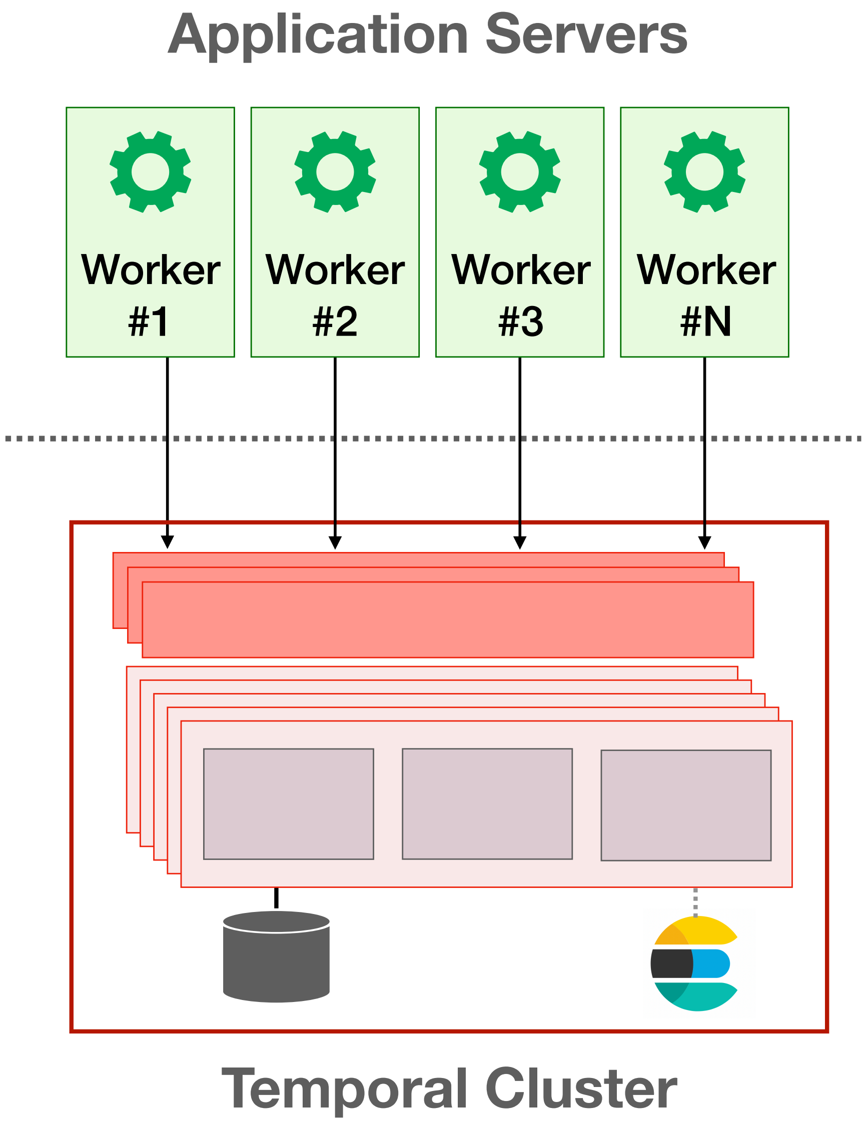
Architectural Overview: Temporal Cluster

- **Temporal Cluster is a complete system**
  - It is a deployment of the Temporal Server software and the components used with it
  - A database is a required component
    - Persists Workflow state and Event History
    - Also stores data for durable timers and queues
  - Elasticsearch is an optional component
    - Improves performance when using advanced search capabilities to locate information about specific Workflow Executions



Architectural Overview: Workers

- **Temporal Cluster *does not* execute your code**
    - It *orchestrates* the execution of your code
- ***Workers* execute your code**
    - They are part of your application
    - They coordinate with the Temporal Cluster
    - It's common to run them on multiple servers



Options for Running a Temporal Cluster

- **Self-Hosted**
  - Using Docker Compose is common for development
  - The new `temporal` command provides an even easier way of running a development cluster
  - Production deployments often run on Kubernetes
- **Temporal Cloud**
  - Access to a Temporal Cluster run by experts via our fully-managed cloud service
    - Dependable: 99.9% uptime SLA and 24x7 production support
    - Frees your organization from having to plan, deploy, and operate your own cluster
  - Your application runs on your own infrastructure

Temporal Clusters for Development (1)

- **The exercise environment for this workshop is already set up for you**
  - It uses the GitPod service to deploy a cluster and browser-based development environment
- **I'll briefly explain two ways to set up your own**
  - These are for reference, so you can experiment on your own after this workshop

Temporal Clusters for Development (2)

- **Docker Compose was historically the most popular option**
  - Temporal provides a GitHub repository with various Docker Compose configurations
  - This runs all of the necessary services within Docker containers
  - It requires that you have already installed Docker and Docker Compose

```
$ git clone https://github.com/temporalio/docker-compose.git

$ cd docker-compose

$ docker-compose up
```

Temporal Clusters for Development (2)

- **The new `temporal` CLI is the fast & easy way to run a development cluster**
  - Install this CLI tool (on a Mac; see docs for other systems)

    ```
    $ brew install temporal
    ```

  - Start a development cluster (using default settings)

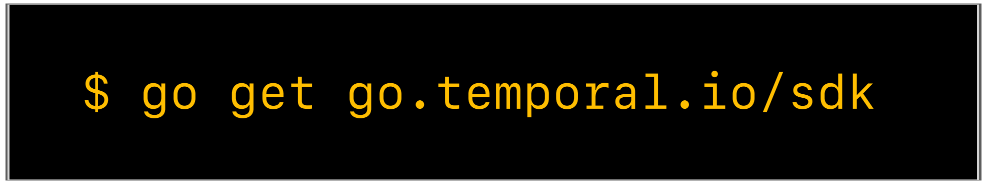
    ```
    $ temporal server start-dev
    ```

  - Start a development cluster (specifying path for durable storage and a custom Web UI port)

    ```
    $ temporal server start-dev \
        --db-filename /Users/twheeler/dev/mycluster.db \
        --ui-port 8080
    ```

Temporal Software Development Kit (SDK)

- **Temporal Workflows are defined in a standard programming language**
  - A Temporal SDK is a language-specific library used to build Temporal applications
  - You will use the APIs it provides when developing Workflows and Worker Programs
  - We currently offer SDKs for several languages

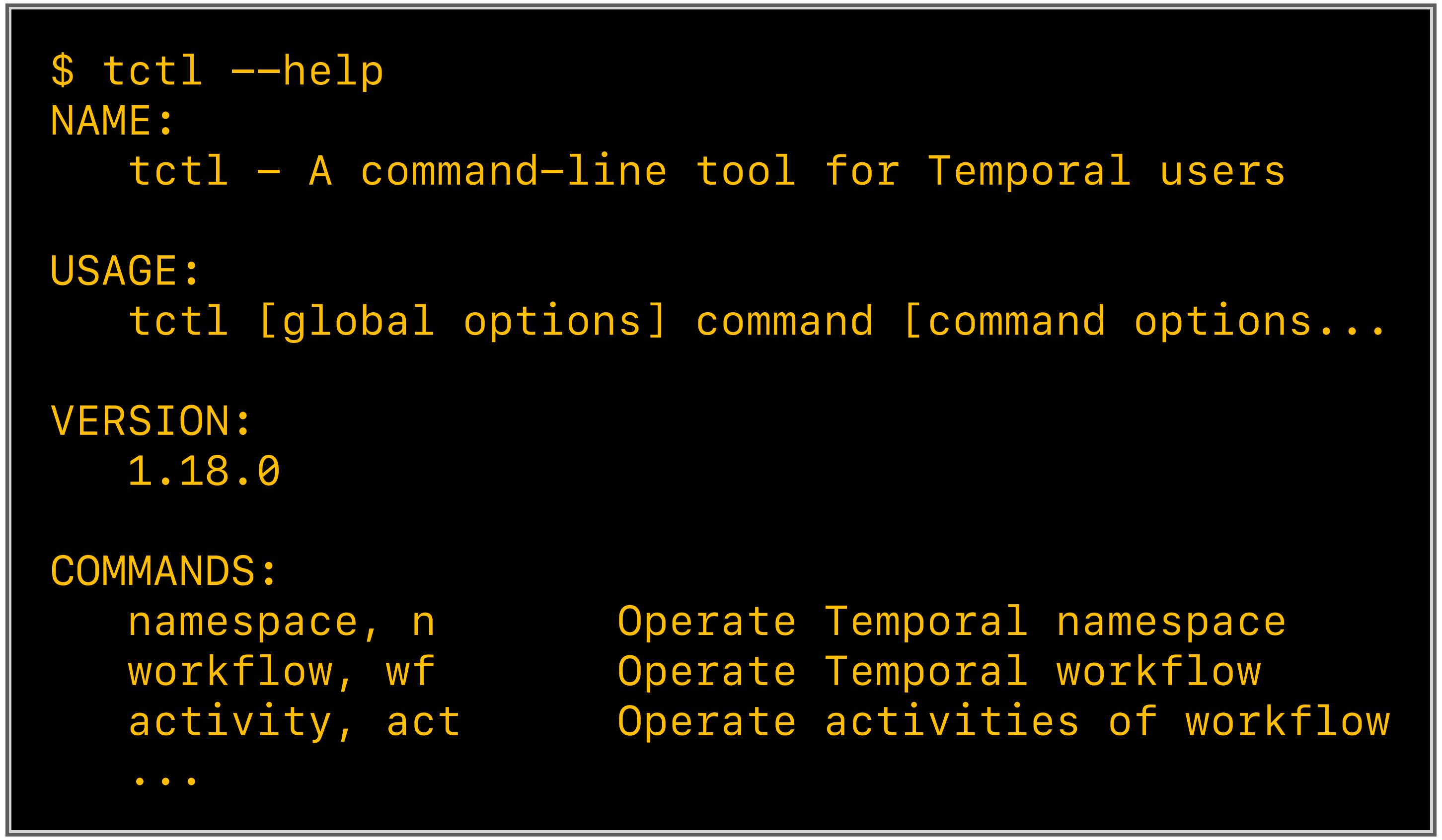


```
$ go get go.temporal.io/sdk
```

This command installs the Temporal SDK for Go

Temporal Command-Line Interface (`tctl`)

- **`tctl` is provides a CLI for interacting with a Temporal cluster**
  - You'll use it to start a Workflow in this workshop, but it has many other capabilities
  - Append --`help` to any command or subcommand to see usage info
  - See documentation for installation instructions
  - This will soon be superseded by the `temporal` command



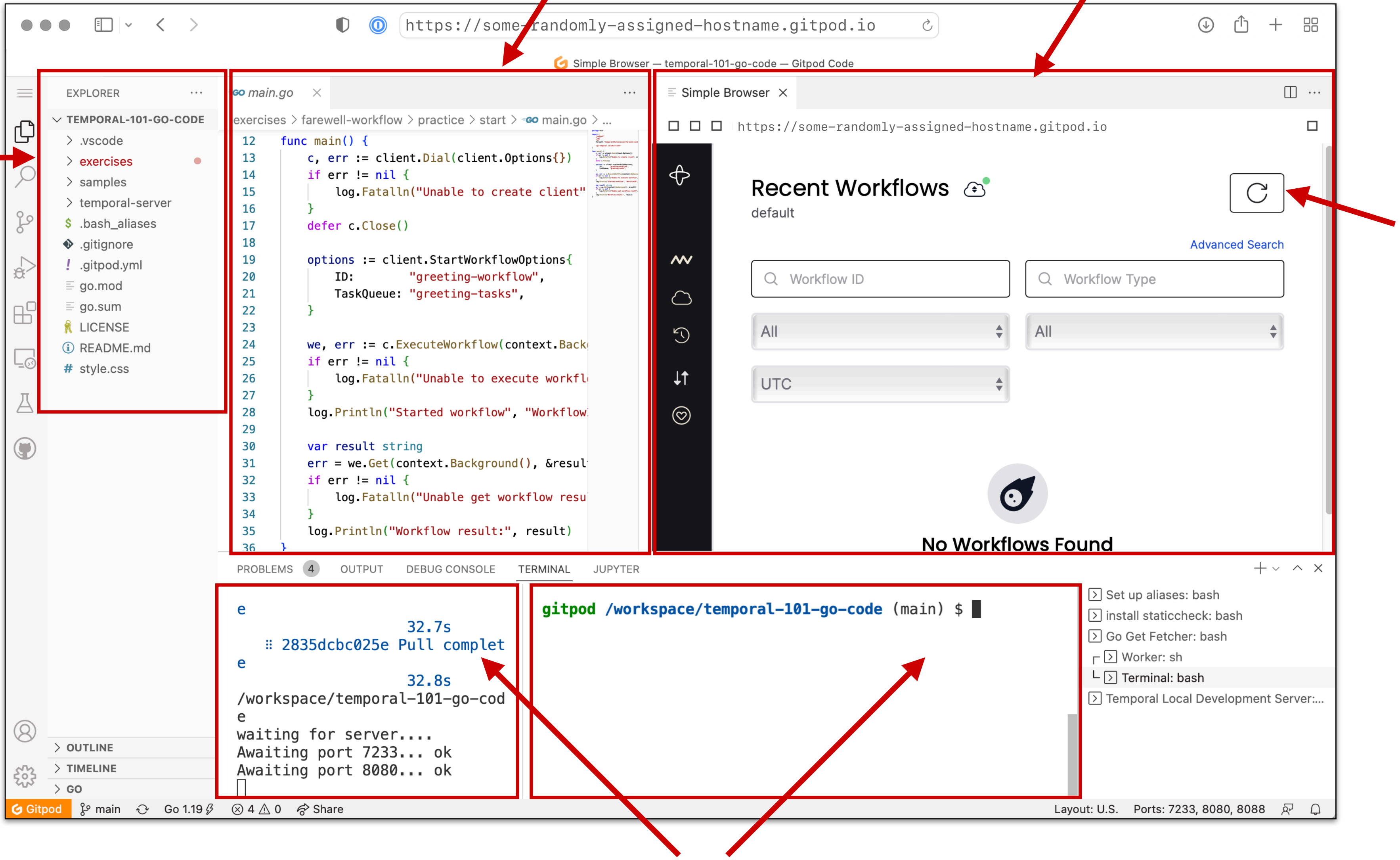
```
$ tctl --help
NAME:
   tctl - A command-line tool for Temporal users

USAGE:
   tctl [global options] command [command options...

VERSION:
   1.18.0

COMMANDS:
   namespace, n        Operate Temporal namespace
   workflow, wf        Operate Temporal workflow
   activity, act       Operate activities of workflow
   ...
```

Exercise Environment

- **We provide a development environment for you in this course**
  - It uses the GitPod service to deploy a private cluster, plus a code editor and terminal
  - You access it through your browser (may require you to log in to GitHub)
  - Your instructor will now demonstrate how to launch and use the environment
    - Please follow along, so your environment will be ready for your first exercise

https://t.mp/replay-101-go-code

GitPod Overview



```go
func main() {
    c, err := client.Dial(client.Options{})
    if err != nil {
        log.Fatalln("Unable to create client"
    }
    defer c.Close()

    options := client.StartWorkflowOptions{
        ID:        "greeting-workflow",
        TaskQueue: "greeting-tasks",
    }

    we, err := c.ExecuteWorkflow(context.Back
    if err != nil {
        log.Fatalln("Unable to execute workfl
    }
    log.Println("Started workflow", "Workflow

    var result string
    err = we.Get(context.Background(), &resul
    if err != nil {
        log.Fatalln("Unable get workflow resu
    }
    log.Println("Workflow result:", result)
}
```

```
e
                        32.7s
   ⠿ 2835dcbc025e Pull complet
e
                        32.8s
/workspace/temporal-101-go-cod
e
waiting for server....
Awaiting port 7233... ok
Awaiting port 8080... ok
```

```
gitpod /workspace/temporal-101-go-code (main) $
```

Code editor

Embedded browser
(displays Temporal Web UI)

File browser
source code for exercises

Refresh button
(for Web UI)

Terminals

Temporal 101

00 About this Workshop

01 What is Temporal?

02 Developing a Workflow

03 Executing a Workflow

04 Viewing Workflow Execution History

05 Modifying an Existing Workflow

06 Developing an Activity

07 Handling Activity Failure

08 Understanding Workflow Execution

09 Conclusion

Business Logic

- **We will begin with an example**
  - Input: string (a person's name)
  - Output: string (a greeting containing that name)
- **This is simply a Go function**
  - It is not (yet) a Temporal Workflow

```go
package app

func GreetSomeone(name string) string {
  return "Hello " + name + "!"
}
```

Executing the Business Logic

- **We can write a small program to invoke that function**
  - Input: string passed on command-line
  - Output: string returned by that function

```go
package main

import(
    "fmt"
    "app"
    "os"
)

func main() {
    name := os.Args[1]
    greeting := app.GreetSomeone(name)
    fmt.Println(greeting)
}
```

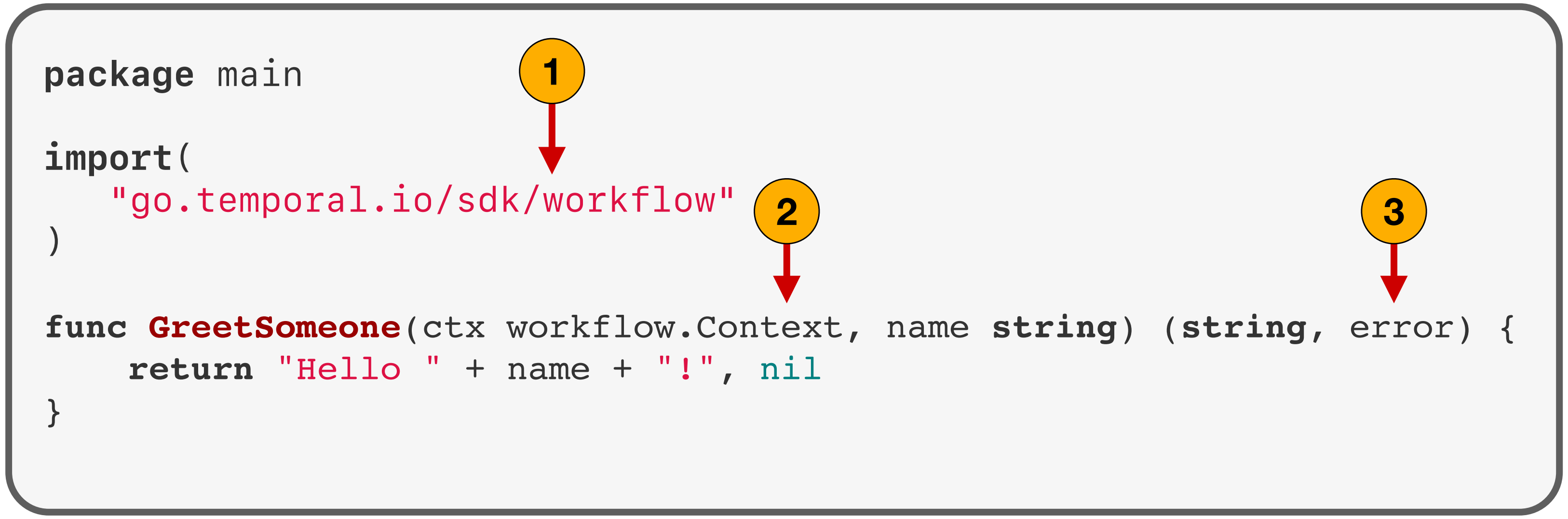
```
$ go run start/main.go Donna
Hello Donna!
```

Workflow Definition

- **With Temporal's Go SDK, you create a Workflow by writing a Go function**
  - The code for this function is known as a *Workflow Definition*
  - Each Workflow has a name, known as its *Workflow Type*
    - In the Go SDK, the Workflow Type is the name of the function (by default)

Writing a Workflow Function

- **Three steps for turning a Go function into a Workflow Definition**
  1. Import the `workflow` package from the SDK
  2. Add `workflow.Context` as the first input parameter
  3. Update the return value to include an error (its value can be `nil`)



```go
package main

import(
    "go.temporal.io/sdk/workflow"
)

func GreetSomeone(ctx workflow.Context, name string) (string, error) {
    return "Hello " + name + "!", nil
}
```

Input Parameters and Return Values

- **Temporal stores the history of your Workflow Executions**
  - Allows you to view input / output of running and completed Workflows
  - Also affects how you will design your Workflows
- **Input parameters and return values must be serializable**
  - Allowed: Null values, binary data, and anything serializable via JSON or Protocol Buffers
  - Prohibited: Channels, functions, and unsafe pointers
- **Avoid passing in or returning large amounts of data from your Workflow**
  - May rapidly expand the size of your Temporal Cluster's database

Initializing the Worker

- **Workers execute your code**

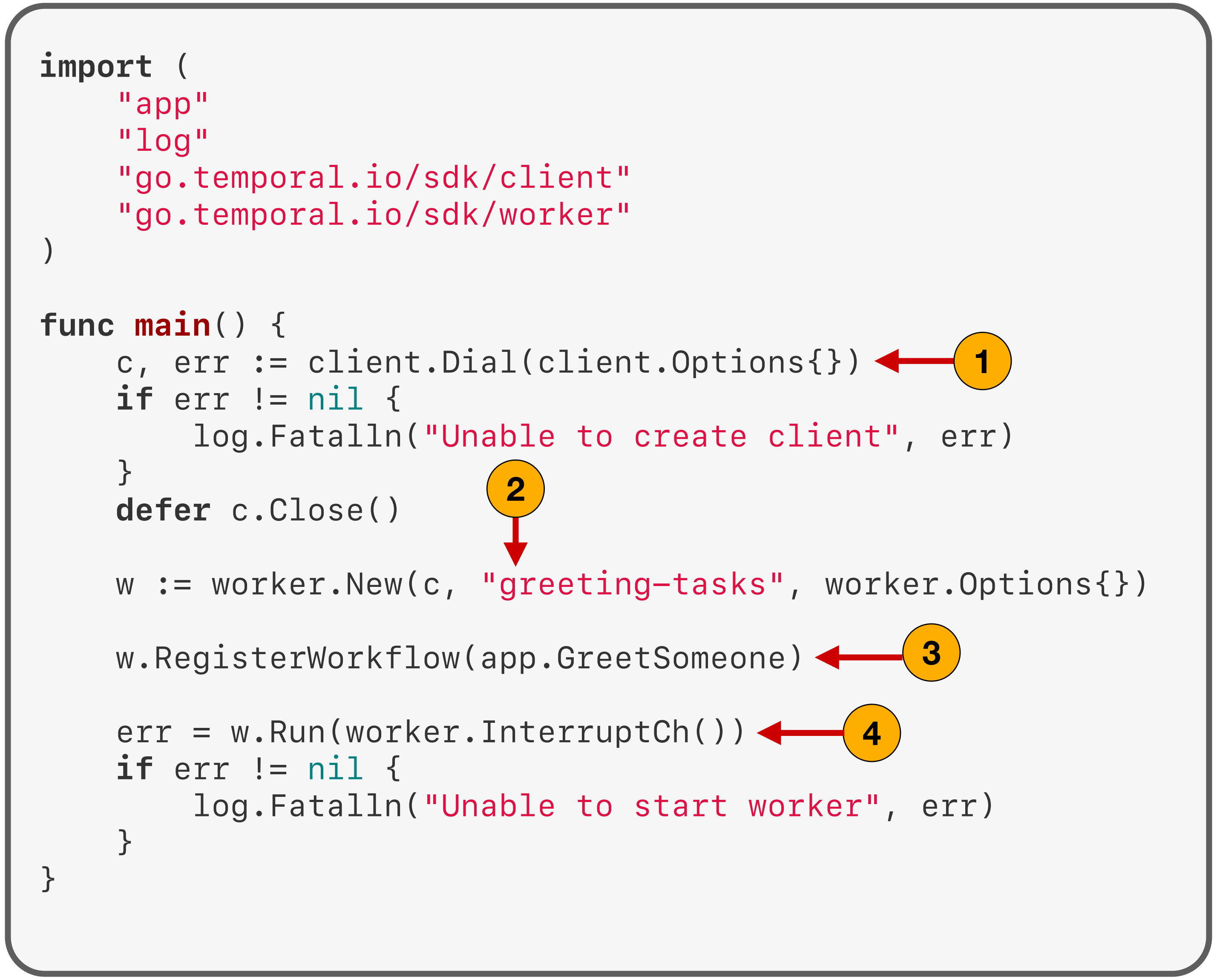
- **How to initialize a Worker**

    1. Configure a Temporal Client, which it uses to communicate with the Temporal Cluster

    2. Specify the name of a task queue on the Temporal Cluster

    3. Register the function(s) it will run

    4. Begin polling the task queue so it can find work to perform



```go
import (
    "app"
    "log"
    "go.temporal.io/sdk/client"
    "go.temporal.io/sdk/worker"
)

func main() {
    c, err := client.Dial(client.Options{})
    if err != nil {
        log.Fatalln("Unable to create client", err)
    }
    defer c.Close()

    w := worker.New(c, "greeting-tasks", worker.Options{})

    w.RegisterWorkflow(app.GreetSomeone)

    err = w.Run(worker.InterruptCh())
    if err != nil {
        log.Fatalln("Unable to start worker", err)
    }
}
```

Temporal 101

00 About this Workshop

01 What is Temporal?

02 Developing a Workflow

03 Executing a Workflow

04 Viewing Workflow Execution History

05 Modifying an Existing Workflow

06 Developing an Activity

07 Handling Activity Failure

08 Understanding Workflow Execution

09 Conclusion

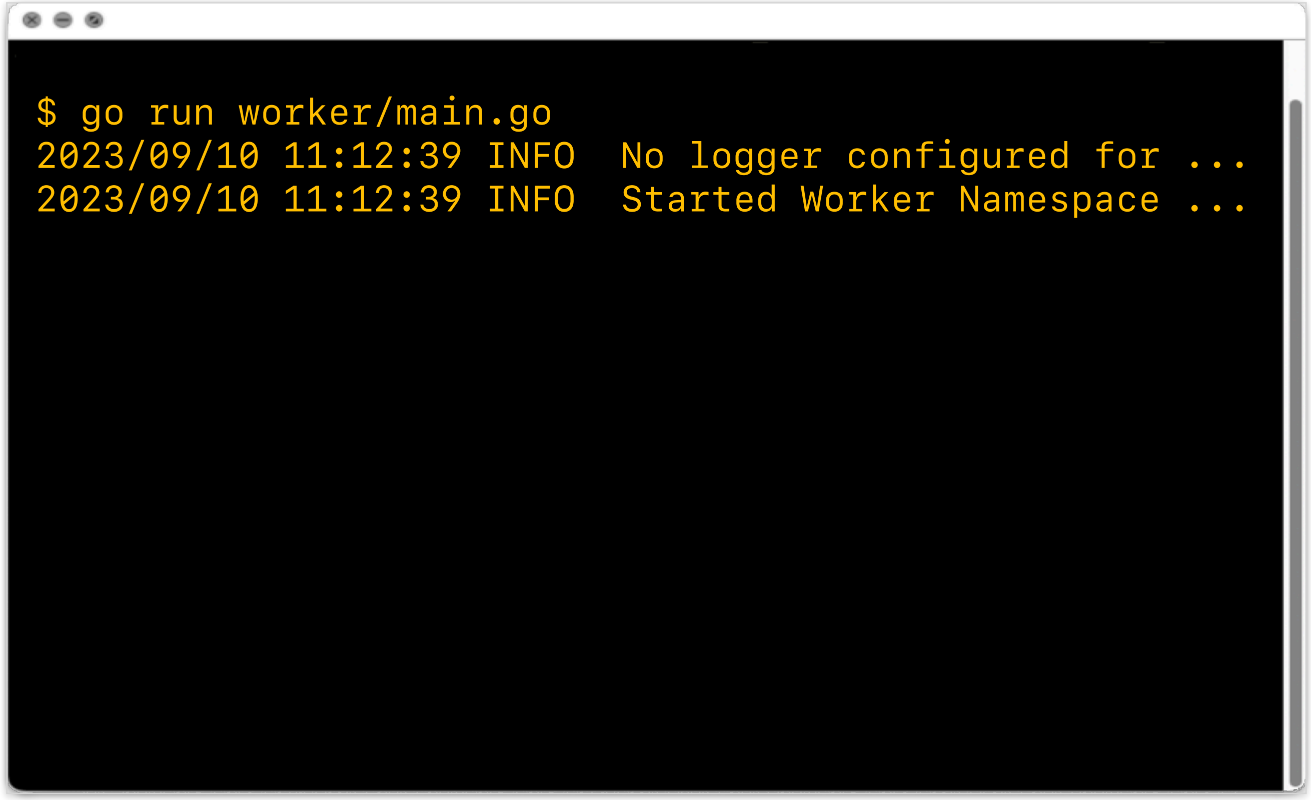
Executing a Workflow from the Command Line

- **One way to start a Workflow is with `tctl workflow start`**
  - The `taskqueue` value must match the value specified in your Worker initialization code
  - The `workflow_id` is a user-defined identifier, which typically has some business meaning
  - The `input` argument's value is unmarshalled and passed as Workflow function parameter

```
$ tctl workflow start \
    --workflow_type GreetSomeone \
    --taskqueue greeting-tasks \
    --workflow_id my-first-workflow \
    --input '"Donna"'

Started Workflow Id: my-first-workflow,
run Id: e8f9217e-344e-4f7b-98bc-7703bc8c7c76
```

Starting the Worker Program

- **Since Workers runs your code, there is no progress unless one is running**
  - After starting it, the Worker program outputs a few lines and then appears to do nothing
  - This is expected behavior, as it is busy polling the task queue and executing your code
  - The Worker will keep running after this Workflow completes, because it then waits for more work to appear in the task queue



```
$ go run worker/main.go
2023/09/10 11:12:39 INFO  No logger configured for ...
2023/09/10 11:12:39 INFO  Started Worker Namespace ...
```

Exercise #1: Hello Workflow

- **During this exercise, you will**
  - Review the business logic of the provided Workflow Definition to understand its behavior
  - Modify the Worker initialization code to specify a task queue name (**`greeting-tasks`**)
  - Run the Worker initialization code to start the Worker process
  - Use **`tctl`** to execute the Workflow from the command line, specifying your name as input
- **Refer to the README.md file in the exercise environment for details**
  - The code is below the **`exercises/hello-workflow`** directory
    - Make your changes to the code in the **`practice`** subdirectory (look for TODO comments)
    - If you need a hint or want to verify your changes, look at the complete version in the **`solution`** subdirectory

Executing a Workflow from Application Code (1)

- **An alternative to using `tctl` is to execute the Workflow from code**
  - This provides a way of integrating Temporal into your own applications
  - You can do this in three steps
    - Import the client package from the SDK
    - Create and configure a client
    - Use the API to request execution
  - We will use similar code to run Workflows in later exercises

```
package main

import(
    "context"
    "log"
    "app"
    "os"
    "go.temporal.io/sdk/client"   1
)

func main() {
    c, err := client.Dial(client.Options{})   2
    if err != nil {
        log.Fatalln("Unable to create client", err)
    }
    defer c.Close()

    // example continues on next slide
```

Executing a Workflow from Application Code (2)

```
    // continued from previous slide

    options := client.StartWorkflowOptions{
       ID:        "my-first-workflow",
       TaskQueue: "greeting-tasks",
    }
                                 3
    we, err := c.ExecuteWorkflow(context.Background(), options, app.GreetSomeone, os.Args[1])
    if err != nil {
        log.Fatalln("Unable to execute workflow", err)
    }
    log.Println("Started workflow", "WorkflowID", we.GetID(), "RunID", we.GetRunID())

    var result string
    err = we.Get(context.Background(), &result)
    if err != nil {
        log.Fatalln("Unable get workflow result", err)
    }
    log.Println("Workflow result:", result)
}
```

Temporal 101

00 About this Workshop

01 What is Temporal?

02 Developing a Workflow

03 Executing a Workflow

04 Viewing Workflow Execution History

05 Modifying an Existing Workflow

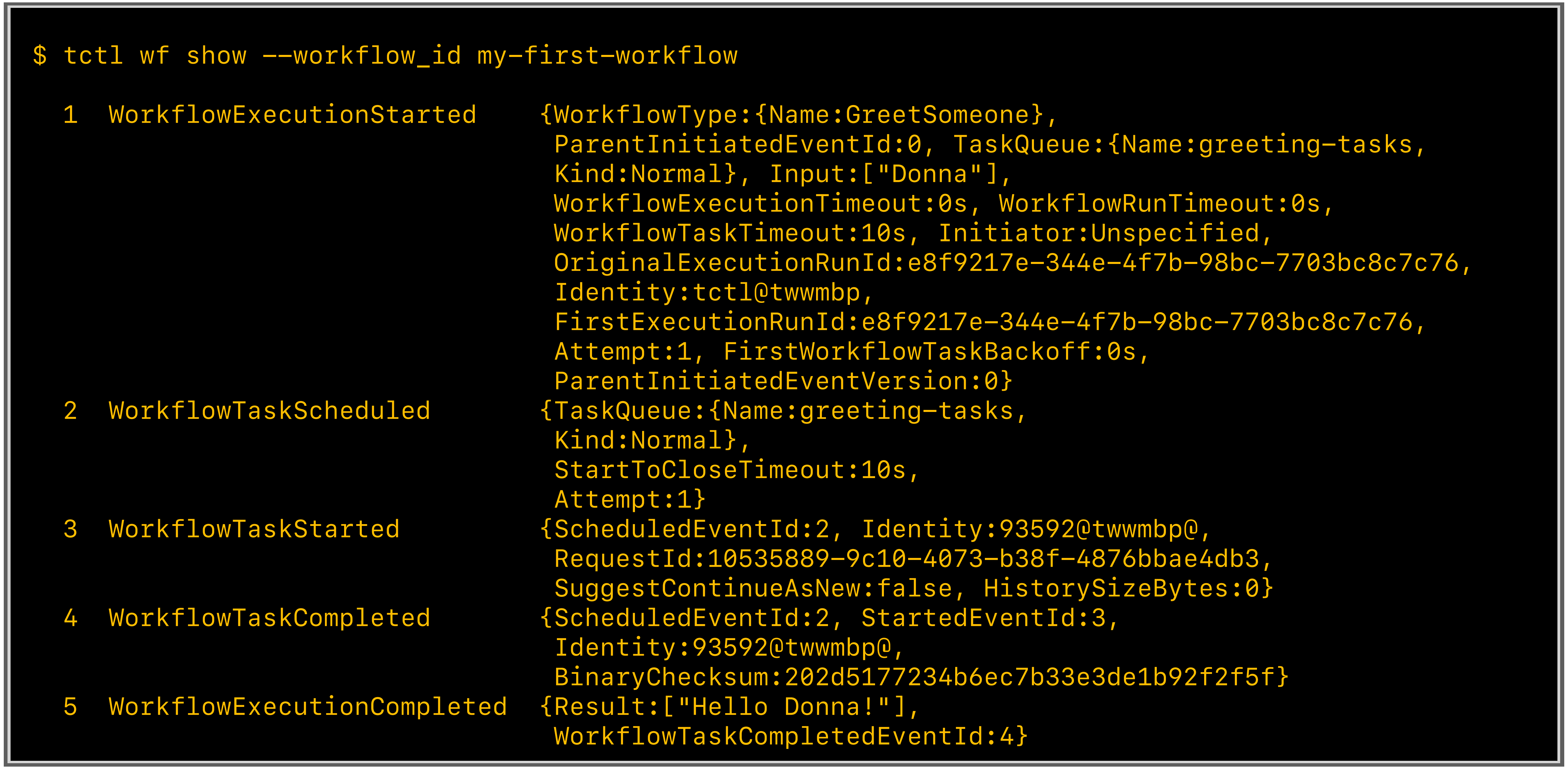
06 Developing an Activity

07 Handling Activity Failure

08 Understanding Workflow Execution

09 Conclusion

Viewing Workflow History with `tctl`

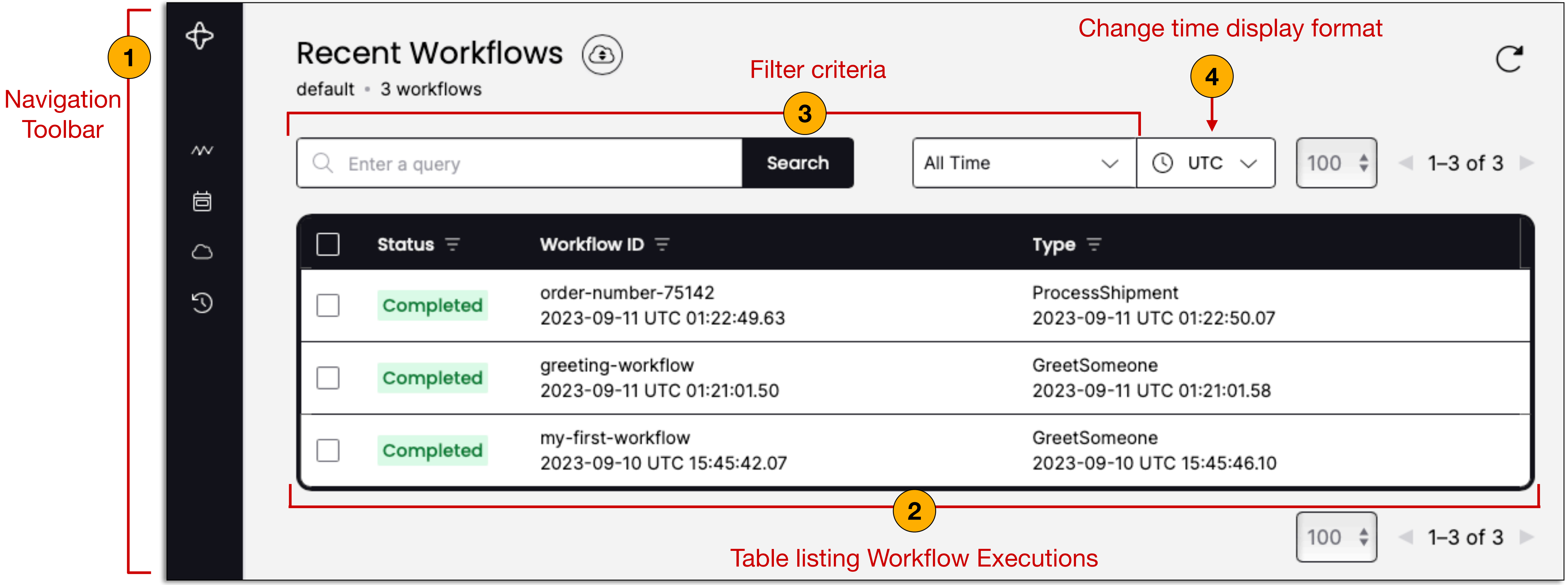


```
$ tctl wf show --workflow_id my-first-workflow

  1  WorkflowExecutionStarted    {WorkflowType:{Name:GreetSomeone},
                                  ParentInitiatedEventId:0, TaskQueue:{Name:greeting-tasks,
                                  Kind:Normal}, Input:["Donna"],
                                  WorkflowExecutionTimeout:0s, WorkflowRunTimeout:0s,
                                  WorkflowTaskTimeout:10s, Initiator:Unspecified,
                                  OriginalExecutionRunId:e8f9217e-344e-4f7b-98bc-7703bc8c7c76,
                                  Identity:tctl@twwmbp,
                                  FirstExecutionRunId:e8f9217e-344e-4f7b-98bc-7703bc8c7c76,
                                  Attempt:1, FirstWorkflowTaskBackoff:0s,
                                  ParentInitiatedEventVersion:0}
  2  WorkflowTaskScheduled       {TaskQueue:{Name:greeting-tasks,
                                  Kind:Normal},
                                  StartToCloseTimeout:10s,
                                  Attempt:1}
  3  WorkflowTaskStarted         {ScheduledEventId:2, Identity:93592@twwmbp@,
                                  RequestId:10535889-9c10-4073-b38f-4876bbae4db3,
                                  SuggestContinueAsNew:false, HistorySizeBytes:0}
  4  WorkflowTaskCompleted       {ScheduledEventId:2, StartedEventId:3,
                                  Identity:93592@twwmbp@,
                                  BinaryChecksum:202d5177234b6ec7b33e3de1b92f2f5f}
  5  WorkflowExecutionCompleted  {Result:["Hello Donna!"],
                                  WorkflowTaskCompletedEventId:4}
```

Viewing History from Web UI

- **The Temporal Web UI displays Workflow status and history**
  - It's also a powerful tool for gaining insight into Workflow Execution
- **The port number used to access it may vary by deployment type**
  - If using Docker Compose on your laptop: `http://localhost:8080/`
  - In our GitPod environment, the Web UI is shown in an embedded browser tab
    - This tab is opened automatically, but there may be a short delay before it's displayed

Web UI: Main Page



Recent Workflows

default • 3 workflows

□	Status	Workflow ID	Type
□	Completed	order-number-75142 2023-09-11 UTC 01:22:49.63	ProcessShipment 2023-09-11 UTC 01:22:50.07
□	Completed	greeting-workflow 2023-09-11 UTC 01:21:01.50	GreetSomeone 2023-09-11 UTC 01:21:01.58
□	Completed	my-first-workflow 2023-09-10 UTC 15:45:42.07	GreetSomeone 2023-09-10 UTC 15:45:46.10

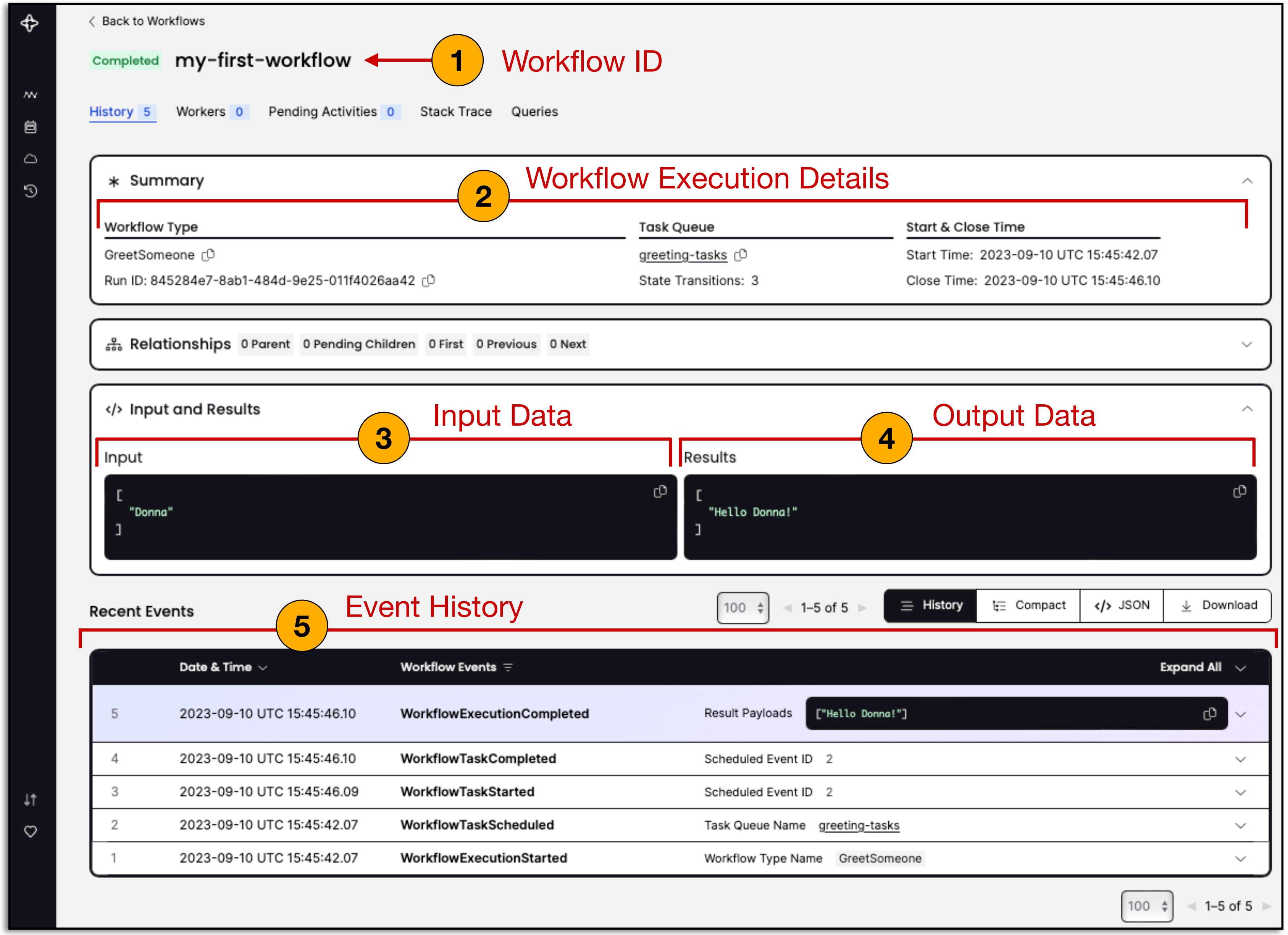
1 Navigation Toolbar

2 Table listing Workflow Executions

3 Filter criteria

4 Change time display format

Web UI: Workflow Execution Detail Page



1 Workflow ID

my-first-workflow

2 Workflow Execution Details

Summary

Workflow Type	Task Queue	Start & Close Time
GreetSomeone	greeting-tasks	Start Time: 2023-09-10 UTC 15:45:42.07
Run ID: 845284e7-8ab1-484d-9e25-011f4026aa42	State Transitions: 3	Close Time: 2023-09-10 UTC 15:45:46.10

Input and Results

3 Input Data

Input

```
[
  "Donna"
]
```

4 Output Data

Results

```
[
  "Hello Donna!"
]
```

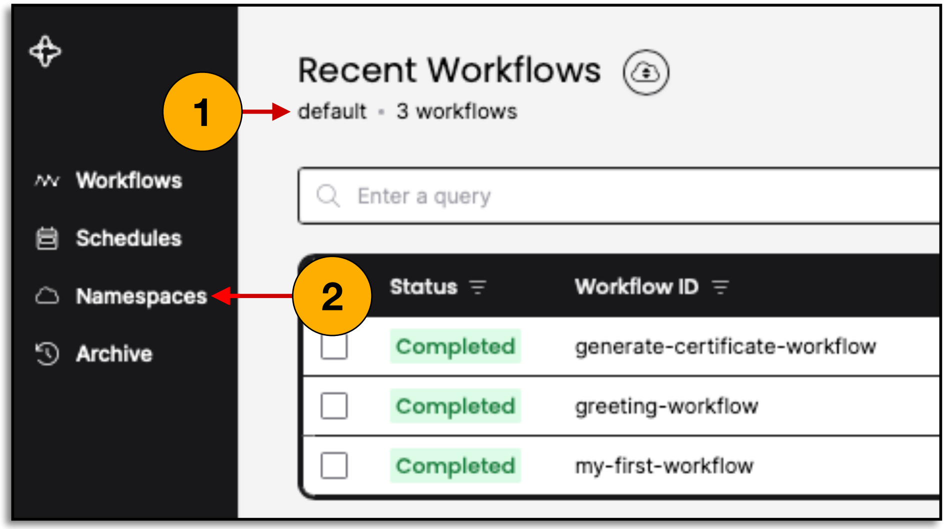
Recent Events

5 Event History

	Date & Time	Workflow Events		
5	2023-09-10 UTC 15:45:46.10	WorkflowExecutionCompleted	Result Payloads	["Hello Donna!"]
4	2023-09-10 UTC 15:45:46.10	WorkflowTaskCompleted	Scheduled Event ID	2
3	2023-09-10 UTC 15:45:46.09	WorkflowTaskStarted	Scheduled Event ID	2
2	2023-09-10 UTC 15:45:42.07	WorkflowTaskScheduled	Task Queue Name	greeting-tasks
1	2023-09-10 UTC 15:45:42.07	WorkflowExecutionStarted	Workflow Type Name	GreetSomeone

Namespaces

- **The Web UI lists recent Workflow Executions within a given *namespace***
  - You can see the selected namespace (1) and switch among available namespaces (2)
- **Namespaces are a means of isolation within a Temporal cluster**
  - Used to logically separate Workflows according to your needs
    - For example, by lifecycle (development vs. production) or department (Marketing vs. Accounting)
  - Some settings are applied at a per-namespace level
  - The default namespace is named `default`


Recent Workflows

default • 3 workflows

Status	Workflow ID
Completed	generate-certificate-workflow
Completed	greeting-workflow
Completed	my-first-workflow

Exercise #2: Hello Web UI

- **During this exercise, you will**
  - Use the Temporal Web UI to display the list of recent Workflow Executions
  - View the detail page for the Workflow Execution from the previous exercise
  - See if you can find the following information on the detail page
    - Name of the task queue
    - Start time
    - Close time (this is the time of completion)
    - Input and output for this Workflow execution (hint: click the "`</> Input and Results`" section)

Temporal 101

00 About this Workshop

01 What is Temporal?

02 Developing a Workflow

03 Executing a Workflow

04 Viewing Workflow Execution History

05 Modifying an Existing Workflow

06 Developing an Activity

07 Handling Activity Failure

08 Understanding Workflow Execution

09 Conclusion

Making Changes to a Workflow

- **Backwards compatibility is an important consideration in Temporal**
- **Avoid changing the number or types of input parameters**
  - We recommend that your Workflow uses a struct as the only input parameter
  - Changing the fields used to create the struct does not change its type
- **You must also ensure that your Workflow is *deterministic***
  - Each execution of a given Workflow must produce the same output, given the same input
  - Tip: You can use Versioning to safely introduce major changes to a Workflow

Restarting the Worker Process

- **Workers use caching for better performance**
  - After making changes, you must restart the Worker(s) before changes take effect
- **The instructor will now demonstrate this**

Temporal 101

00 About this Workshop

01 What is Temporal?

02 Developing a Workflow

03 Executing a Workflow

04 Viewing Workflow Execution History

05 Modifying an Existing Workflow

06 Developing an Activity

07 Handling Activity Failure

08 Understanding Workflow Execution

09 Conclusion

What Are Activities?

- **Activities encapsulate business logic that is prone to failure**
  - They are executed during Workflow Execution
  - If an Activity fails, it will be retried
- **Activity Definitions are Go functions**
  - Rules for input and output types are the same as for Workflow Definitions
  - Temporal does not impose a naming convention on the function name
  - Does not have to be in same source file as Workflow, but can be if you prefer
  - Although not required, we recommend passing `context.Context` as the first parameter

Registering Activities

- **Like Workflows, Activities must also be registered with the Worker**
  - The process is similar, too

```go
func main() {
  c, err := client.Dial(client.Options{})
  if err != nil {
    log.Fatalln("Unable to create client", err)
  }
  defer c.Close()

  w := worker.New(c, "greeting-tasks", worker.Options{})

  w.RegisterWorkflow(app.GreetSomeone)
  w.RegisterActivity(app.GreetInSpanish)

  err = w.Run(worker.InterruptCh())
  if err != nil {
    log.Fatalln("Unable to start worker", err)
  }
}
```

Executing Activities

```go
package app

import (
  "go.temporal.io/sdk/workflow"
  "time"
)

func GreetSomeone(ctx workflow.Context, name string) (string, error) {
  options := workflow.ActivityOptions{
    StartToCloseTimeout: time.Second * 5,
  }
  ctx = workflow.WithActivityOptions(ctx, options)

  var spanishGreeting string
  err := workflow.ExecuteActivity(ctx, GreetInSpanish, name).Get(ctx, &spanishGreeting)
  if err != nil {
    return "", err
  }

  return spanishGreeting, nil
}
```

Temporal 101

00 About this Workshop

01 What is Temporal?

02 Developing a Workflow

03 Executing a Workflow

04 Viewing Workflow Execution History

05 Modifying an Existing Workflow

06 Developing an Activity

07 Handling Activity Failure

08 Understanding Workflow Execution

09 Conclusion

How Temporal Handles Activity Failure

- **By default, Temporal automatically retries failed Activities forever**
- **Four properties determine the timing and number of retry attempts**
  - You can override one or more of these defaults with a custom Retry Policy

Property	Description	Default Value
`InitialInterval`	Duration before the first retry	1 second
`BackoffCoefficient`	Multiplier used for subsequent retries	`2.0`
`MaximumInterval`	Maximum duration between retries	`100 * InitialInterval`
`MaximumAttempts`	Maximum number of retry attempts before giving up	`0` (unlimited)

Activity Retry Policy Example

```go
import (
    "go.temporal.io/sdk/workflow"
    "go.temporal.io/sdk/temporal"
    "time"
)

func GreetSomeone(ctx workflow.Context, name string) (string, error) {
    retrypolicy := &temporal.RetryPolicy {
      InitialInterval:    15 * time.Second,  // first retry will occur after 15 seconds
      BackoffCoefficient: 2.0,               // double the delay after each retry
      MaximumInterval:    time.Second * 60,  // up to a maximum delay of 60 seconds
      MaximumAttempts:    100,               // fail the Activity after 100 attempts
    }

    options := workflow.ActivityOptions {
        StartToCloseTimeout: time.Second * 5,
        RetryPolicy: retrypolicy,
    }
    ctx = workflow.WithActivityOptions(ctx, options)

    var result string
    err := workflow.ExecuteActivity(ctx, GreetInSpanish, name).Get(ctx, &result)

    // ... remainder of Workflow code would follow
```

1 Import this package from the SDK

2 Specify your policy values

3 Associate the policy with the Activity options

Exercise #3: Farewell Workflow

- **During this exercise, you will**
  - Write an Activity function
  - Register the Activity function
  - Modify the Workflow to execute your new Activity
  - Run the Workflow
- **Refer to the README.md file in the exercise environment for details**
  - The code is below the **`exercises/farewell-workflow`** directory
    - Make your changes to the code in the **`practice`** subdirectory (look for TODO comments)
    - If you need a hint or want to verify your changes, look at the complete version in the **`solution`** subdirectory

Temporal 101

00 About this Workshop

01 What is Temporal?

02 Developing a Workflow

03 Executing a Workflow

04 Viewing Workflow Execution History

05 Modifying an Existing Workflow

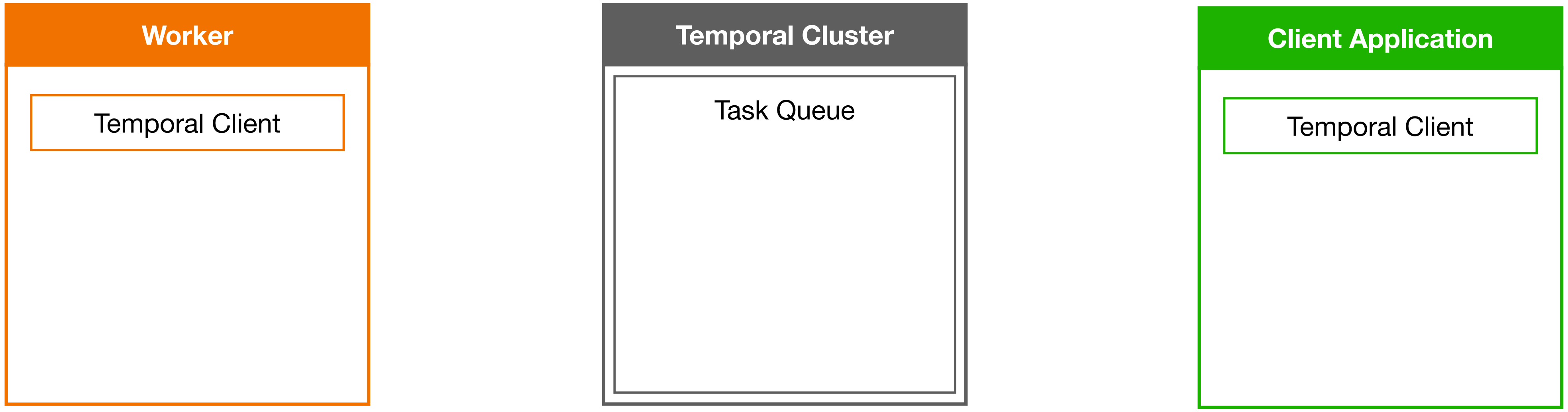
06 Developing an Activity

07 Handling Activity Failure

08 Understanding Workflow Execution

09 Conclusion

Actors in this Workflow Execution Scenario



Worker

Temporal Client

Temporal Cluster

Task Queue

Client Application

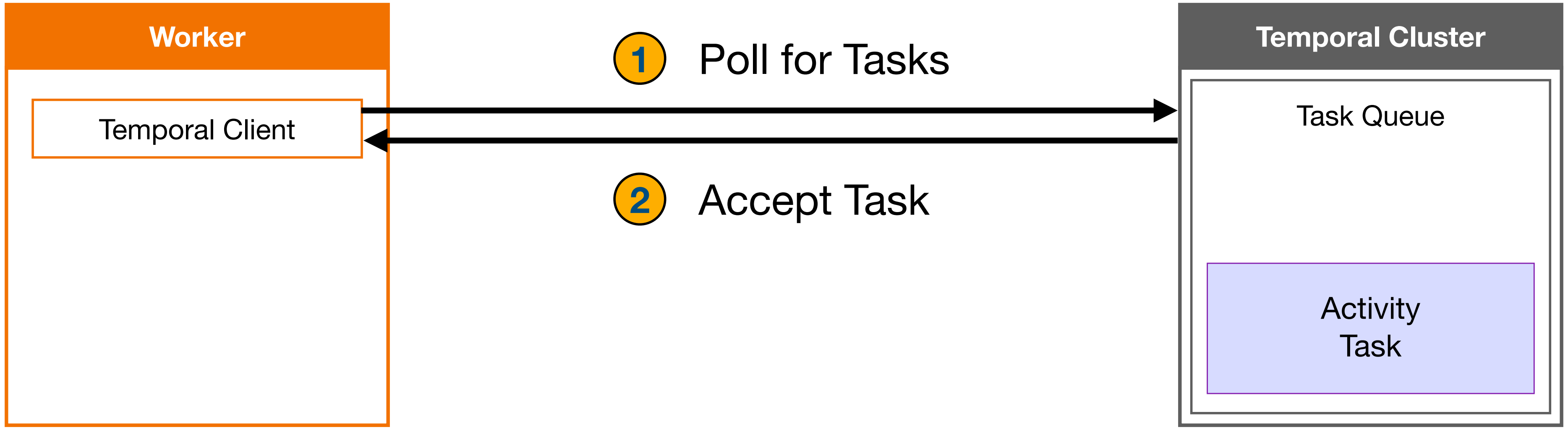
Temporal Client

Executes the code

Orchestrates code execution

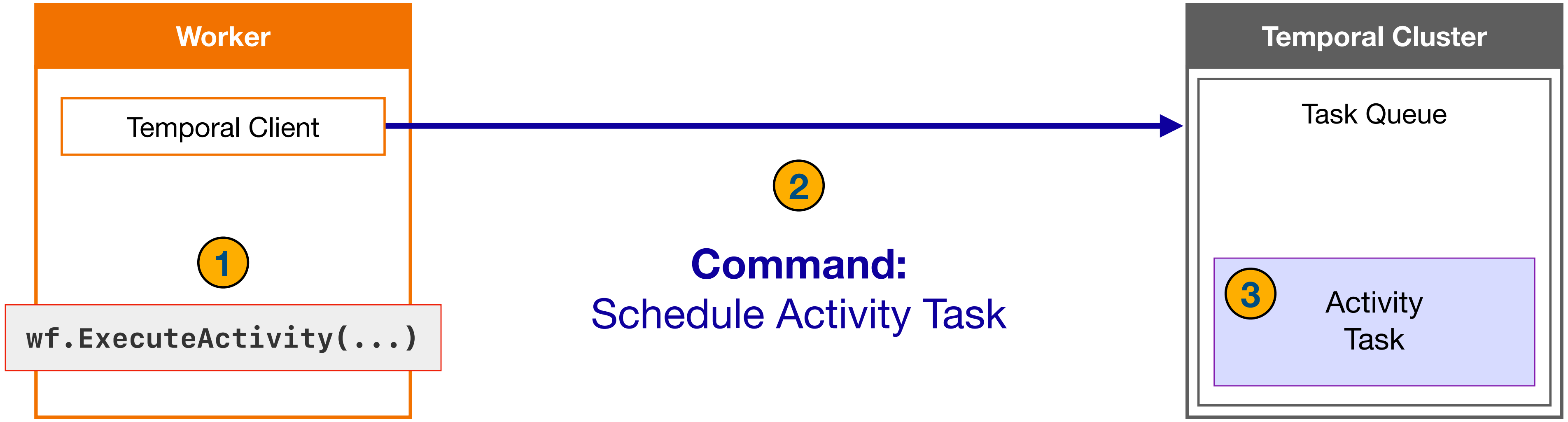
Requests code execution and retrieves the result

Workers and Tasks



- Temporal does not assign tasks to Workers
- Workers continuously poll, accepting tasks when they have spare capacity
- You can increase throughput and scalability by adding Workers

Commands

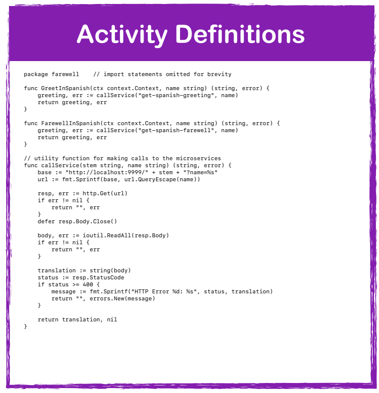


- Certain API calls result in the Worker issuing a Command to the Temporal Cluster
- The Cluster acts on these commands, but also stores them
- This allows the Worker to recreate the state of a Workflow Execution following a crash

Activity Definitions

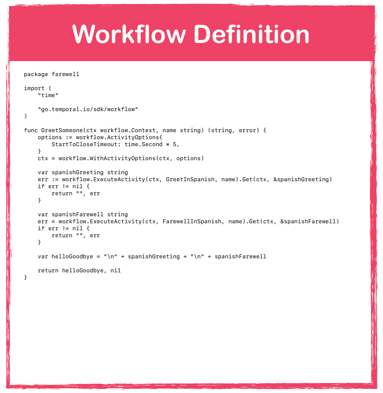
```go
package farewell    // import statements omitted for brevity

func GreetInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-greeting", name)
    return greeting, err
}

func FarewellInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-farewell", name)
    return greeting, err
}

// utility function for making calls to the microservices
func callService(stem string, name string) (string, error) {
    base := "http://localhost:9999/" + stem + "?name=%s"
    url := fmt.Sprintf(base, url.QueryEscape(name))

    resp, err := http.Get(url)
    if err != nil {
        return "", err
    }
    defer resp.Body.Close()

    body, err := ioutil.ReadAll(resp.Body)
    if err != nil {
        return "", err
    }

    translation := string(body)
    status := resp.StatusCode
    if status >= 400 {
        message := fmt.Sprintf("HTTP Error %d: %s", status, translation)
        return "", errors.New(message)
    }

    return translation, nil
}
```

This is just a utility function for calling the microservice and is not specific to Temporal



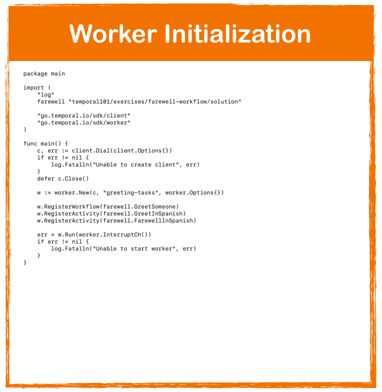
Activity Definitions

```go
package farewell    // import statements omitted for brevity

func GreetInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-greeting", name)
    return greeting, err
}

func FarewellInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-farewell", name)
    return greeting, err
}

// utility function for making calls to the microservices
func callService(stem string, name string) (string, error) {
    base := "http://localhost:9999/" + stem + "?name=%s"
    url := fmt.Sprintf(base, url.QueryEscape(name))

    resp, err := http.Get(url)
    if err != nil {
        return "", err
    }
    defer resp.Body.Close()

    body, err := ioutil.ReadAll(resp.Body)
    if err != nil {
        return "", err
    }

    translation := string(body)
    status := resp.StatusCode
    if status >= 400 {
        message := fmt.Sprintf("HTTP Error %d: %s", status, translation)
        return "", errors.New(message)
    }

    return translation, nil
}
```



Workflow Definition

```go
package farewell

import (
    "time"

    "go.temporal.io/sdk/workflow"
)

func GreetSomeone(ctx workflow.Context, name string) (string, error) {
    options := workflow.ActivityOptions{
        StartToCloseTimeout: time.Second * 5,
    }
    ctx = workflow.WithActivityOptions(ctx, options)

    var spanishGreeting string
    err := workflow.ExecuteActivity(ctx, GreetInSpanish, name).Get(ctx, &spanishGreeting)
    if err != nil {
        return "", err
    }

    var spanishFarewell string
    err = workflow.ExecuteActivity(ctx, FarewellInSpanish, name).Get(ctx, &spanishFarewell)
    if err != nil {
        return "", err
    }

    var helloGoodbye = "\n" + spanishGreeting + "\n" + spanishFarewell

    return helloGoodbye, nil
}
```



Worker Initialization

```go
package main

import (
    "log"
    farewell "temporal101/exercises/farewell-workflow/solution"

    "go.temporal.io/sdk/client"
    "go.temporal.io/sdk/worker"
)

func main() {
    c, err := client.Dial(client.Options{})
    if err != nil {
        log.Fatalln("Unable to create client", err)
    }
    defer c.Close()

    w := worker.New(c, "greeting-tasks", worker.Options{})

    w.RegisterWorkflow(farewell.GreetSomeone)
    w.RegisterActivity(farewell.GreetInSpanish)
    w.RegisterActivity(farewell.FarewellInSpanish)

    err = w.Run(worker.InterruptCh())
    if err != nil {
        log.Fatalln("Unable to start worker", err)
    }
}
```

Workflow Definition

Activity Definitions

```go
package farewell    // import statements omitted for brevity

func GreetInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-greeting", name)
    return greeting, err
}

func FarewellInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-farewell", name)
    return greeting, err
}

// utility function for making calls to the microservices
func callService(stem string, name string) (string, error) {
    base := "http://localhost:9999/" + stem + "?name=%s"
    url := fmt.Sprintf(base, url.QueryEscape(name))

    resp, err := http.Get(url)
    if err != nil {
        return "", err
    }
    defer resp.Body.Close()

    body, err := ioutil.ReadAll(resp.Body)
    if err != nil {
        return "", err
    }

    translation := string(body)
    status := resp.StatusCode
    if status >= 400 {
        message := fmt.Sprintf("HTTP Error %d: %s", status, translation)
        return "", errors.New(message)
    }

    return translation, nil
}
```

Workflow Definition

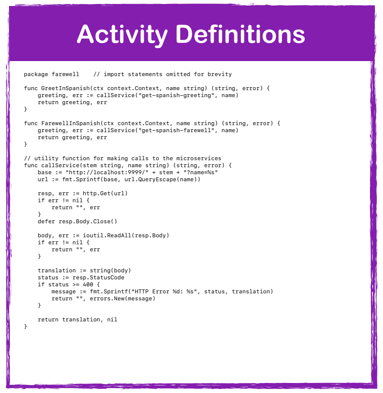
```go
package farewell

import (
    "time"

    "go.temporal.io/sdk/workflow"
)

func GreetSomeone(ctx workflow.Context, name string) (string, error) {
    options := workflow.ActivityOptions{
        StartToCloseTimeout: time.Second * 5,
    }
    ctx = workflow.WithActivityOptions(ctx, options)

    var spanishGreeting string
    err := workflow.ExecuteActivity(ctx, GreetInSpanish, name).Get(ctx, &spanishGreeting)
    if err != nil {
        return "", err
    }

    var spanishFarewell string
    err = workflow.ExecuteActivity(ctx, FarewellInSpanish, name).Get(ctx, &spanishFarewell)
    if err != nil {
        return "", err
    }

    var helloGoodbye = "\n" + spanishGreeting + "\n" + spanishFarewell

    return helloGoodbye, nil
}
```

Worker Initialization

```go
package main

import (
    "log"
    farewell "temporal101/exercises/farewell-workflow/solution"

    "go.temporal.io/sdk/client"
    "go.temporal.io/sdk/worker"
)

func main() {
    c, err := client.Dial(client.Options{})
    if err != nil {
        log.Fatalln("Unable to create client", err)
    }
    defer c.Close()

    w := worker.New(c, "greeting-tasks", worker.Options{})

    w.RegisterWorkflow(farewell.GreetSomeone)
    w.RegisterActivity(farewell.GreetInSpanish)
    w.RegisterActivity(farewell.FarewellInSpanish)

    err = w.Run(worker.InterruptCh())
    if err != nil {
        log.Fatalln("Unable to start worker", err)
    }
}
```

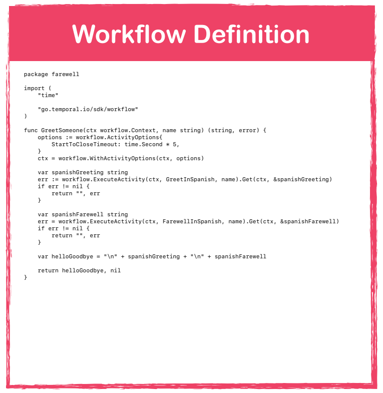
```go
package farewell

import (
    "time"

    "go.temporal.io/sdk/workflow"
)

func GreetSomeone(ctx workflow.Context, name string) (string, error) {
    options := workflow.ActivityOptions{
        StartToCloseTimeout: time.Second * 5,
    }
    ctx = workflow.WithActivityOptions(ctx, options)

    var spanishGreeting string
    err := workflow.ExecuteActivity(ctx, GreetInSpanish, name).Get(ctx, &spanishGreeting)
    if err != nil {
        return "", err
    }

    var spanishFarewell string
    err = workflow.ExecuteActivity(ctx, FarewellInSpanish, name).Get(ctx, &spanishFarewell)
    if err != nil {
        return "", err
    }

    var helloGoodbye = "\n" + spanishGreeting + "\n" + spanishFarewell

    return helloGoodbye, nil
}
```

Worker Initialization



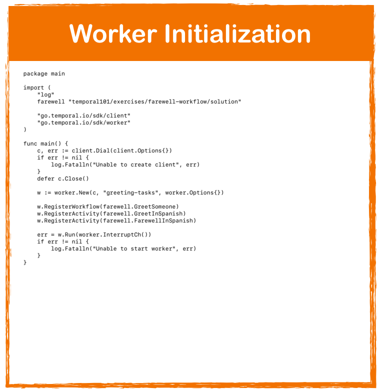
Activity Definitions

```go
package farewell    // import statements omitted for brevity

func GreetInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-greeting", name)
    return greeting, err
}

func FarewellInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-farewell", name)
    return greeting, err
}

// utility function for making calls to the microservices
func callService(stem string, name string) (string, error) {
    base := "http://localhost:9999/" + stem + "?name=%s"
    url := fmt.Sprintf(base, url.QueryEscape(name))

    resp, err := http.Get(url)
    if err != nil {
        return "", err
    }
    defer resp.Body.Close()

    body, err := ioutil.ReadAll(resp.Body)
    if err != nil {
        return "", err
    }

    translation := string(body)
    status := resp.StatusCode
    if status >= 400 {
        message := fmt.Sprintf("HTTP Error %d: %s", status, translation)
        return "", errors.New(message)
    }

    return translation, nil
}
```



Workflow Definition

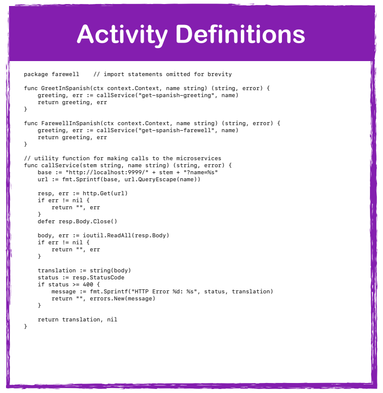
```go
package farewell

import (
    "time"

    "go.temporal.io/sdk/workflow"
)

func GreetSomeone(ctx workflow.Context, name string) (string, error) {
    options := workflow.ActivityOptions{
        StartToCloseTimeout: time.Second * 5,
    }
    ctx = workflow.WithActivityOptions(ctx, options)

    var spanishGreeting string
    err := workflow.ExecuteActivity(ctx, GreetInSpanish, name).Get(ctx, &spanishGreeting)
    if err != nil {
        return "", err
    }

    var spanishFarewell string
    err = workflow.ExecuteActivity(ctx, FarewellInSpanish, name).Get(ctx, &spanishFarewell)
    if err != nil {
        return "", err
    }

    var helloGoodbye = "\n" + spanishGreeting + "\n" + spanishFarewell

    return helloGoodbye, nil
}
```



Worker Initialization

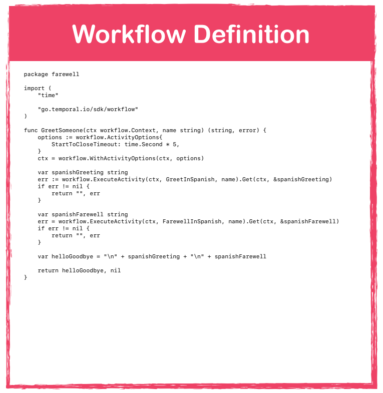
```go
package main

import (
    "log"
    farewell "temporal101/exercises/farewell-workflow/solution"

    "go.temporal.io/sdk/client"
    "go.temporal.io/sdk/worker"
)

func main() {
    c, err := client.Dial(client.Options{})
    if err != nil {
        log.Fatalln("Unable to create client", err)
    }
    defer c.Close()

    w := worker.New(c, "greeting-tasks", worker.Options{})

    w.RegisterWorkflow(farewell.GreetSomeone)
    w.RegisterActivity(farewell.GreetInSpanish)
    w.RegisterActivity(farewell.FarewellInSpanish)

    err = w.Run(worker.InterruptCh())
    if err != nil {
        log.Fatalln("Unable to start worker", err)
    }
}
```

```go
package main

import (
    "log"
    farewell "temporal101/exercises/farewell-workflow/solution"

    "go.temporal.io/sdk/client"
    "go.temporal.io/sdk/worker"
)

func main() {
    c, err := client.Dial(client.Options{})
    if err != nil {
        log.Fatalln("Unable to create client", err)
    }
    defer c.Close()

    w := worker.New(c, "greeting-tasks", worker.Options{})

    w.RegisterWorkflow(farewell.GreetSomeone)
    w.RegisterActivity(farewell.GreetInSpanish)
    w.RegisterActivity(farewell.FarewellInSpanish)

    err = w.Run(worker.InterruptCh())
    if err != nil {
        log.Fatalln("Unable to start worker", err)
    }
}
```



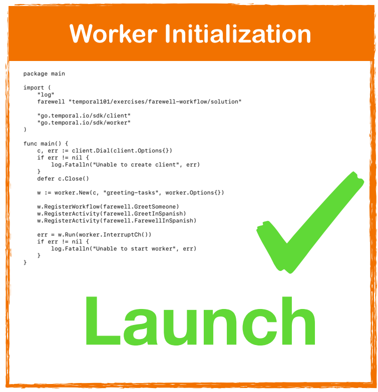
Activity Definitions

```go
package farewell    // import statements omitted for brevity

func GreetInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-greeting", name)
    return greeting, err
}

func FarewellInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-farewell", name)
    return greeting, err
}

// utility function for making calls to the microservices
func callService(stem string, name string) (string, error) {
    base := "http://localhost:9999/" + stem + "?name=%s"
    url := fmt.Sprintf(base, url.QueryEscape(name))

    resp, err := http.Get(url)
    if err != nil {
        return "", err
    }
    defer resp.Body.Close()

    body, err := ioutil.ReadAll(resp.Body)
    if err != nil {
        return "", err
    }

    translation := string(body)
    status := resp.StatusCode
    if status >= 400 {
        message := fmt.Sprintf("HTTP Error %d: %s", status, translation)
        return "", errors.New(message)
    }

    return translation, nil
}
```



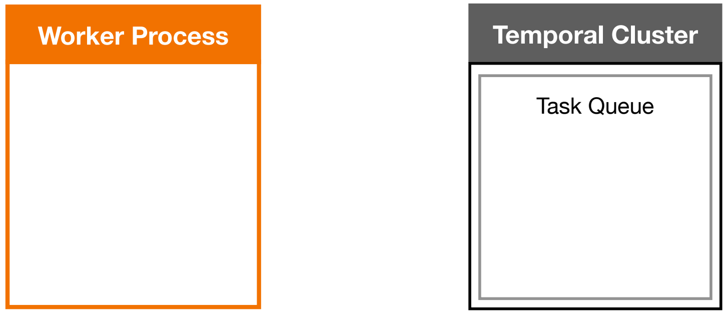
Workflow Definition

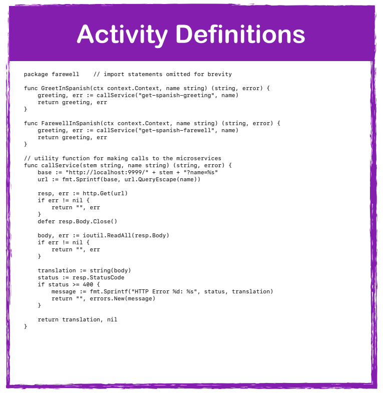
```go
package farewell

import (
    "time"

    "go.temporal.io/sdk/workflow"
)

func GreetSomeone(ctx workflow.Context, name string) (string, error) {
    options := workflow.ActivityOptions{
        StartToCloseTimeout: time.Second * 5,
    }
    ctx = workflow.WithActivityOptions(ctx, options)

    var spanishGreeting string
    err := workflow.ExecuteActivity(ctx, GreetInSpanish, name).Get(ctx, &spanishGreeting)
    if err != nil {
        return "", err
    }

    var spanishFarewell string
    err = workflow.ExecuteActivity(ctx, FarewellInSpanish, name).Get(ctx, &spanishFarewell)
    if err != nil {
        return "", err
    }

    var helloGoodbye = "\n" + spanishGreeting + "\n" + spanishFarewell

    return helloGoodbye, nil
}
```



Worker Initialization

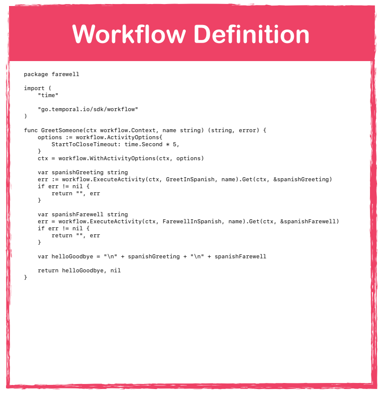
```go
package main

import (
    "log"
    farewell "temporal101/exercises/farewell-workflow/solution"

    "go.temporal.io/sdk/client"
    "go.temporal.io/sdk/worker"
)

func main() {
    c, err := client.Dial(client.Options{})
    if err != nil {
        log.Fatalln("Unable to create client", err)
    }
    defer c.Close()

    w := worker.New(c, "greeting-tasks", worker.Options{})

    w.RegisterWorkflow(farewell.GreetSomeone)
    w.RegisterActivity(farewell.GreetInSpanish)
    w.RegisterActivity(farewell.FarewellInSpanish)

    err = w.Run(worker.InterruptCh())
    if err != nil {
        log.Fatalln("Unable to start worker", err)
    }
}
```





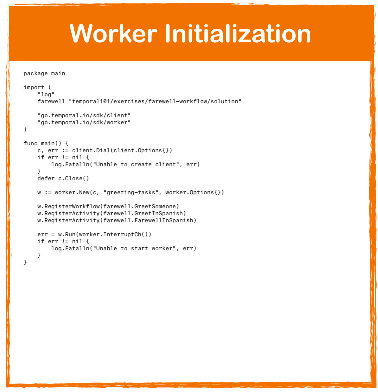
Activity Definitions

```go
package farewell    // import statements omitted for brevity

func GreetInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-greeting", name)
    return greeting, err
}

func FarewellInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-farewell", name)
    return greeting, err
}

// utility function for making calls to the microservices
func callService(stem string, name string) (string, error) {
    base := "http://localhost:9999/" + stem + "?name=%s"
    url := fmt.Sprintf(base, url.QueryEscape(name))

    resp, err := http.Get(url)
    if err != nil {
        return "", err
    }
    defer resp.Body.Close()

    body, err := ioutil.ReadAll(resp.Body)
    if err != nil {
        return "", err
    }

    translation := string(body)
    status := resp.StatusCode
    if status >= 400 {
        message := fmt.Sprintf("HTTP Error %d: %s", status, translation)
        return "", errors.New(message)
    }

    return translation, nil
}
```



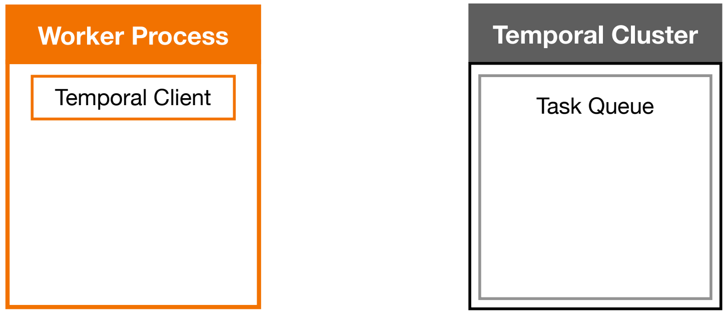
Workflow Definition

```go
package farewell

import (
    "time"

    "go.temporal.io/sdk/workflow"
)

func GreetSomeone(ctx workflow.Context, name string) (string, error) {
    options := workflow.ActivityOptions{
        StartToCloseTimeout: time.Second * 5,
    }
    ctx = workflow.WithActivityOptions(ctx, options)

    var spanishGreeting string
    err := workflow.ExecuteActivity(ctx, GreetInSpanish, name).Get(ctx, &spanishGreeting)
    if err != nil {
        return "", err
    }

    var spanishFarewell string
    err = workflow.ExecuteActivity(ctx, FarewellInSpanish, name).Get(ctx, &spanishFarewell)
    if err != nil {
        return "", err
    }

    var helloGoodbye = "\n" + spanishGreeting + "\n" + spanishFarewell

    return helloGoodbye, nil
}
```



Worker Initialization

```go
package main

import (
    "log"
    farewell "temporal101/exercises/farewell-workflow/solution"

    "go.temporal.io/sdk/client"
    "go.temporal.io/sdk/worker"
)

func main() {
    c, err := client.Dial(client.Options{})
    if err != nil {
        log.Fatalln("Unable to create client", err)
    }
    defer c.Close()

    w := worker.New(c, "greeting-tasks", worker.Options{})

    w.RegisterWorkflow(farewell.GreetSomeone)
    w.RegisterActivity(farewell.GreetInSpanish)
    w.RegisterActivity(farewell.FarewellInSpanish)

    err = w.Run(worker.InterruptCh())
    if err != nil {
        log.Fatalln("Unable to start worker", err)
    }
}
```



Activity Definitions

```go
package farewell    // import statements omitted for brevity

func GreetInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-greeting", name)
    return greeting, err
}

func FarewellInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-farewell", name)
    return greeting, err
}

// utility function for making calls to the microservices
func callService(stem string, name string) (string, error) {
    base := "http://localhost:9999/" + stem + "?name=%s"
    url := fmt.Sprintf(base, url.QueryEscape(name))

    resp, err := http.Get(url)
    if err != nil {
        return "", err
    }
    defer resp.Body.Close()

    body, err := ioutil.ReadAll(resp.Body)
    if err != nil {
        return "", err
    }

    translation := string(body)
    status := resp.StatusCode
    if status >= 400 {
        message := fmt.Sprintf("HTTP Error %d: %s", status, translation)
        return "", errors.New(message)
    }

    return translation, nil
}
```

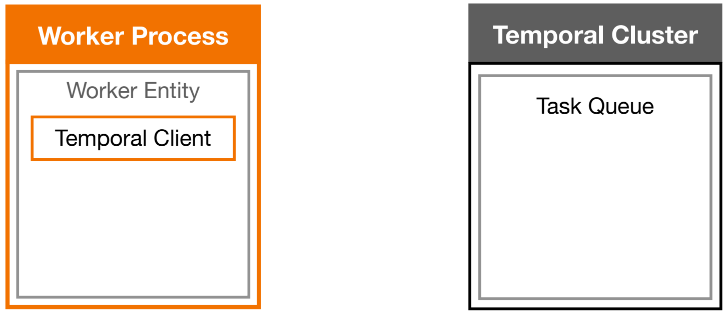
Workflow Definition

```go
package farewell

import (
    "time"

    "go.temporal.io/sdk/workflow"
)

func GreetSomeone(ctx workflow.Context, name string) (string, error) {
    options := workflow.ActivityOptions{
        StartToCloseTimeout: time.Second * 5,
    }
    ctx = workflow.WithActivityOptions(ctx, options)

    var spanishGreeting string
    err := workflow.ExecuteActivity(ctx, GreetInSpanish, name).Get(ctx, &spanishGreeting)
    if err != nil {
        return "", err
    }

    var spanishFarewell string
    err = workflow.ExecuteActivity(ctx, FarewellInSpanish, name).Get(ctx, &spanishFarewell)
    if err != nil {
        return "", err
    }

    var helloGoodbye = "\n" + spanishGreeting + "\n" + spanishFarewell

    return helloGoodbye, nil
}
```

Worker Initialization

```go
package main

import (
    "log"
    farewell "temporal101/exercises/farewell-workflow/solution"

    "go.temporal.io/sdk/client"
    "go.temporal.io/sdk/worker"
)

func main() {
    c, err := client.Dial(client.Options{})
    if err != nil {
        log.Fatalln("Unable to create client", err)
    }
    defer c.Close()

    w := worker.New(c, "greeting-tasks", worker.Options{})

    w.RegisterWorkflow(farewell.GreetSomeone)
    w.RegisterActivity(farewell.GreetInSpanish)
    w.RegisterActivity(farewell.FarewellInSpanish)

    err = w.Run(worker.InterruptCh())
    if err != nil {
        log.Fatalln("Unable to start worker", err)
    }
}
```

```go
package main

import (
    "log"
    farewell "temporal101/exercises/farewell-workflow/solution"

    "go.temporal.io/sdk/client"
    "go.temporal.io/sdk/worker"
)

func main() {
    c, err := client.Dial(client.Options{})
    if err != nil {
        log.Fatalln("Unable to create client", err)
    }
    defer c.Close()

    w := worker.New(c, "greeting-tasks", worker.Options{})

    w.RegisterWorkflow(farewell.GreetSomeone)
    w.RegisterActivity(farewell.GreetInSpanish)
    w.RegisterActivity(farewell.FarewellInSpanish)

    err = w.Run(worker.InterruptCh())
    if err != nil {
        log.Fatalln("Unable to start worker", err)
    }
}
```



Activity Definitions

```
package farewell    // import statements omitted for brevity

func GreetInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-greeting", name)
    return greeting, err
}

func FarewellInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-farewell", name)
    return greeting, err
}

// utility function for making calls to the microservices
func callService(stem string, name string) (string, error) {
    base := "http://localhost:9999/" + stem + "?name=%s"
    url := fmt.Sprintf(base, url.QueryEscape(name))

    resp, err := http.Get(url)
    if err != nil {
        return "", err
    }
    defer resp.Body.Close()

    body, err := ioutil.ReadAll(resp.Body)
    if err != nil {
        return "", err
    }

    translation := string(body)
    status := resp.StatusCode
    if status >= 400 {
        message := fmt.Sprintf("HTTP Error %d: %s", status, translation)
        return "", errors.New(message)
    }

    return translation, nil
}
```

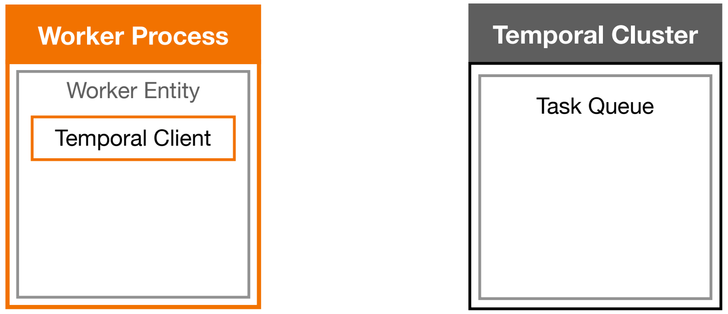
Workflow Definition

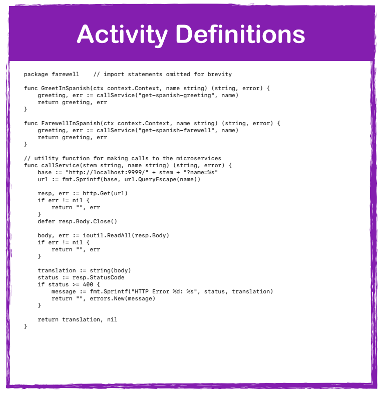
```
package farewell

import (
    "time"

    "go.temporal.io/sdk/workflow"
)

func GreetSomeone(ctx workflow.Context, name string) (string, error) {
    options := workflow.ActivityOptions{
        StartToCloseTimeout: time.Second * 5,
    }
    ctx = workflow.WithActivityOptions(ctx, options)

    var spanishGreeting string
    err := workflow.ExecuteActivity(ctx, GreetInSpanish, name).Get(ctx, &spanishGreeting)
    if err != nil {
        return "", err
    }

    var spanishFarewell string
    err = workflow.ExecuteActivity(ctx, FarewellInSpanish, name).Get(ctx, &spanishFarewell)
    if err != nil {
        return "", err
    }

    var helloGoodbye = "\n" + spanishGreeting + "\n" + spanishFarewell

    return helloGoodbye, nil
}
```

Worker Initialization

```
package main

import (
    "log"
    farewell "temporal101/exercises/farewell-workflow/solution"

    "go.temporal.io/sdk/client"
    "go.temporal.io/sdk/worker"
)

func main() {
    c, err := client.Dial(client.Options{})
    if err != nil {
        log.Fatalln("Unable to create client", err)
    }
    defer c.Close()

    w := worker.New(c, "greeting-tasks", worker.Options{})

    w.RegisterWorkflow(farewell.GreetSomeone)
    w.RegisterActivity(farewell.GreetInSpanish)
    w.RegisterActivity(farewell.FarewellInSpanish)

    err = w.Run(worker.InterruptCh())
    if err != nil {
        log.Fatalln("Unable to start worker", err)
    }
}
```

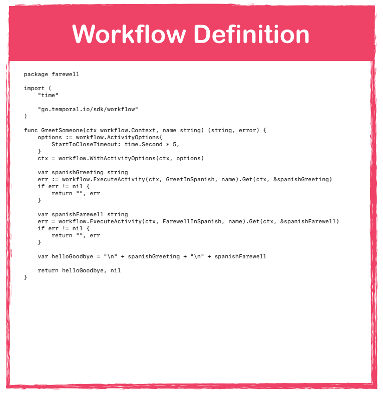
```
package main

import (
    "log"
    farewell "temporal101/exercises/farewell-workflow/solution"

    "go.temporal.io/sdk/client"
    "go.temporal.io/sdk/worker"
)

func main() {
    c, err := client.Dial(client.Options{})
    if err != nil {
        log.Fatalln("Unable to create client", err)
    }
    defer c.Close()

    w := worker.New(c, "greeting-tasks", worker.Options{})

    w.RegisterWorkflow(farewell.GreetSomeone)
    w.RegisterActivity(farewell.GreetInSpanish)
    w.RegisterActivity(farewell.FarewellInSpanish)

    err = w.Run(worker.InterruptCh())
    if err != nil {
        log.Fatalln("Unable to start worker", err)
    }
}
```





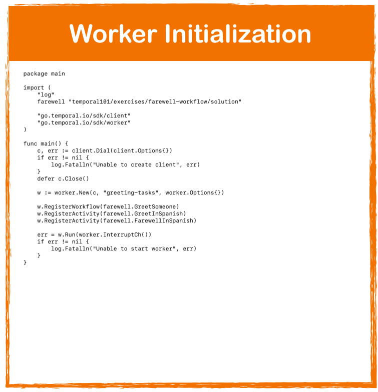
Activity Definitions

```
package farewell    // import statements omitted for brevity

func GreetInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-greeting", name)
    return greeting, err
}

func FarewellInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-farewell", name)
    return greeting, err
}

// utility function for making calls to the microservices
func callService(stem string, name string) (string, error) {
    base := "http://localhost:9999/" + stem + "?name=%s"
    url := fmt.Sprintf(base, url.QueryEscape(name))

    resp, err := http.Get(url)
    if err != nil {
        return "", err
    }
    defer resp.Body.Close()

    body, err := ioutil.ReadAll(resp.Body)
    if err != nil {
        return "", err
    }

    translation := string(body)
    status := resp.StatusCode
    if status >= 400 {
        message := fmt.Sprintf("HTTP Error %d: %s", status, translation)
        return "", errors.New(message)
    }

    return translation, nil
}
```



Workflow Definition

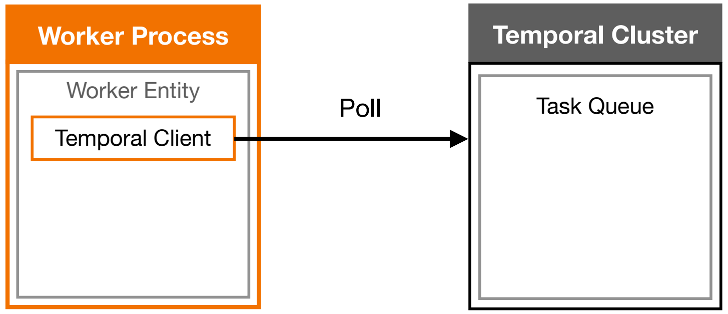
```
package farewell

import (
    "time"

    "go.temporal.io/sdk/workflow"
)

func GreetSomeone(ctx workflow.Context, name string) (string, error) {
    options := workflow.ActivityOptions{
        StartToCloseTimeout: time.Second * 5,
    }
    ctx = workflow.WithActivityOptions(ctx, options)

    var spanishGreeting string
    err := workflow.ExecuteActivity(ctx, GreetInSpanish, name).Get(ctx, &spanishGreeting)
    if err != nil {
        return "", err
    }

    var spanishFarewell string
    err = workflow.ExecuteActivity(ctx, FarewellInSpanish, name).Get(ctx, &spanishFarewell)
    if err != nil {
        return "", err
    }

    var helloGoodbye = "\n" + spanishGreeting + "\n" + spanishFarewell

    return helloGoodbye, nil
}
```



Worker Initialization

```
package main

import (
    "log"
    farewell "temporal101/exercises/farewell-workflow/solution"

    "go.temporal.io/sdk/client"
    "go.temporal.io/sdk/worker"
)

func main() {
    c, err := client.Dial(client.Options{})
    if err != nil {
        log.Fatalln("Unable to create client", err)
    }
    defer c.Close()

    w := worker.New(c, "greeting-tasks", worker.Options{})

    w.RegisterWorkflow(farewell.GreetSomeone)
    w.RegisterActivity(farewell.GreetInSpanish)
    w.RegisterActivity(farewell.FarewellInSpanish)

    err = w.Run(worker.InterruptCh())
    if err != nil {
        log.Fatalln("Unable to start worker", err)
    }
}
```

```
package main

import (
    "log"
    farewell "temporal101/exercises/farewell-workflow/solution"

    "go.temporal.io/sdk/client"
    "go.temporal.io/sdk/worker"
)

func main() {
    c, err := client.Dial(client.Options{})
    if err != nil {
        log.Fatalln("Unable to create client", err)
    }
    defer c.Close()

    w := worker.New(c, "greeting-tasks", worker.Options{})

    w.RegisterWorkflow(farewell.GreetSomeone)
    w.RegisterActivity(farewell.GreetInSpanish)
    w.RegisterActivity(farewell.FarewellInSpanish)

    err = w.Run(worker.InterruptCh())
    if err != nil {
        log.Fatalln("Unable to start worker", err)
    }
}
```



Launching from Command Line

Activity Definitions

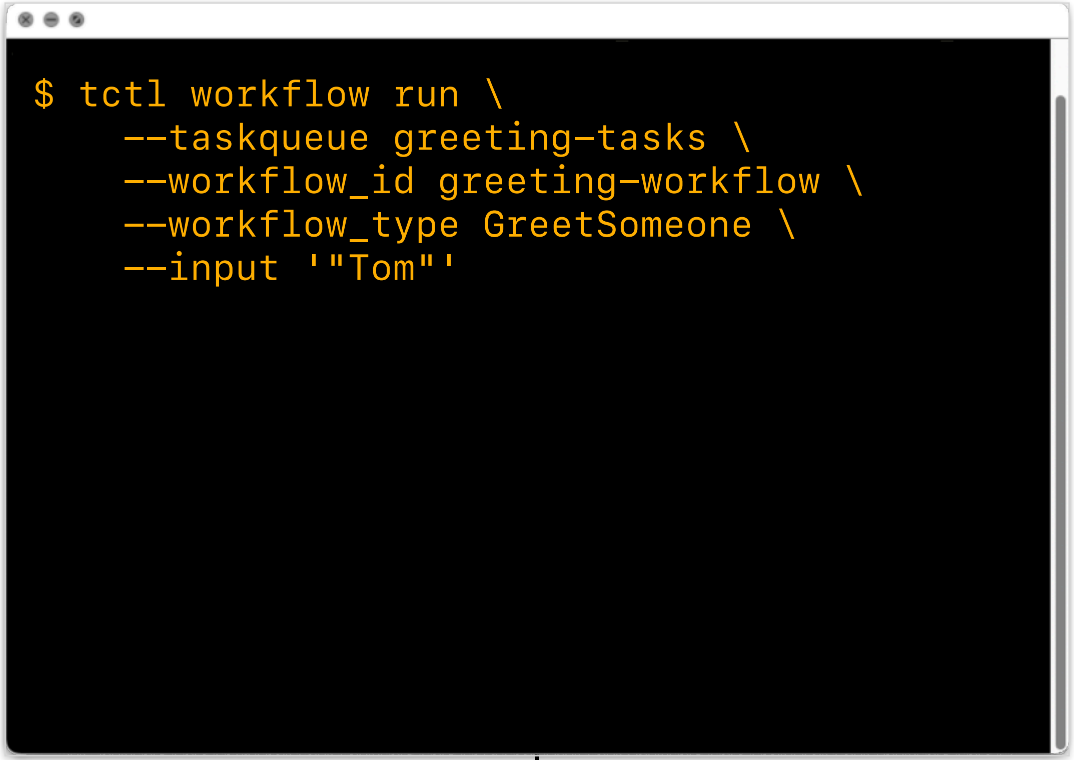
```go
package farewell    // import statements omitted for brevity

func GreetInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-greeting", name)
    return greeting, err
}

func FarewellInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-farewell", name)
    return greeting, err
}

// utility function for making calls to the microservices
func callService(stem string, name string) (string, error) {
    base := "http://localhost:9999/" + stem + "?name=%s"
    url := fmt.Sprintf(base, url.QueryEscape(name))

    resp, err := http.Get(url)
    if err != nil {
        return "", err
    }
    defer resp.Body.Close()

    body, err := ioutil.ReadAll(resp.Body)
    if err != nil {
        return "", err
    }

    translation := string(body)
    status := resp.StatusCode
    if status >= 400 {
        message := fmt.Sprintf("HTTP Error %d: %s", status, translation)
        return "", errors.New(message)
    }

    return translation, nil
}
```

Workflow Definition

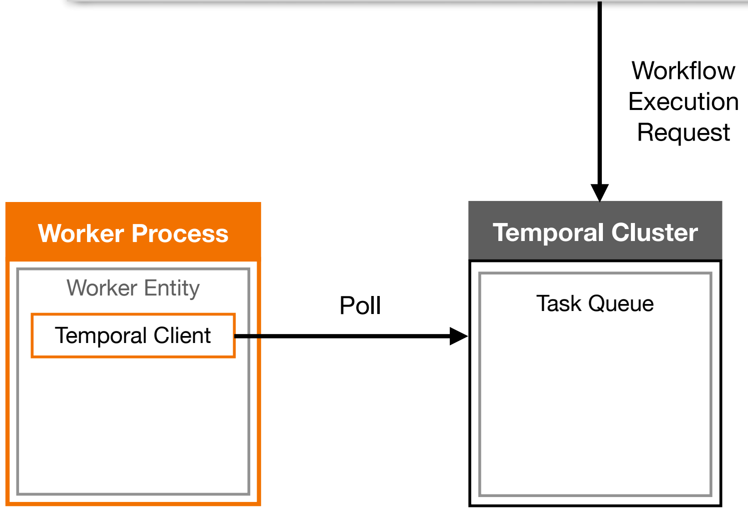
```go
package farewell

import (
    "time"

    "go.temporal.io/sdk/workflow"
)

func GreetSomeone(ctx workflow.Context, name string) (string, error) {
    options := workflow.ActivityOptions{
        StartToCloseTimeout: time.Second * 5,
    }
    ctx = workflow.WithActivityOptions(ctx, options)

    var spanishGreeting string
    err := workflow.ExecuteActivity(ctx, GreetInSpanish, name).Get(ctx, &spanishGreeting)
    if err != nil {
        return "", err
    }

    var spanishFarewell string
    err = workflow.ExecuteActivity(ctx, FarewellInSpanish, name).Get(ctx, &spanishFarewell)
    if err != nil {
        return "", err
    }

    var helloGoodbye = "\n" + spanishGreeting + "\n" + spanishFarewell

    return helloGoodbye, nil
}
```

Worker Initialization

```go
package main

import (
    "log"
    farewell "temporal101/exercises/farewell-workflow/solution"

    "go.temporal.io/sdk/client"
    "go.temporal.io/sdk/worker"
)

func main() {
    c, err := client.Dial(client.Options{})
    if err != nil {
        log.Fatalln("Unable to create client", err)
    }
    defer c.Close()

    w := worker.New(c, "greeting-tasks", worker.Options{})

    w.RegisterWorkflow(farewell.GreetSomeone)
    w.RegisterActivity(farewell.GreetInSpanish)
    w.RegisterActivity(farewell.FarewellInSpanish)

    err = w.Run(worker.InterruptCh())
    if err != nil {
        log.Fatalln("Unable to start worker", err)
    }
}
```



```
$ tctl workflow run \
    --taskqueue greeting-tasks \
    --workflow_id greeting-workflow \
    --workflow_type GreetSomeone \
    --input '"Tom"'
```



Launching from Application Code

Activity Definitions

```go
package farewell    // import statements omitted for brevity

func GreetInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-greeting", name)
    return greeting, err
}

func FarewellInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-farewell", name)
    return greeting, err
}

// utility function for making calls to the microservices
func callService(stem string, name string) (string, error) {
    base := "http://localhost:9999/" + stem + "?name=%s"
    url := fmt.Sprintf(base, url.QueryEscape(name))

    resp, err := http.Get(url)
    if err != nil {
        return "", err
    }
    defer resp.Body.Close()

    body, err := ioutil.ReadAll(resp.Body)
    if err != nil {
        return "", err
    }

    translation := string(body)
    status := resp.StatusCode
    if status >= 400 {
        message := fmt.Sprintf("HTTP Error %d: %s", status, translation)
        return "", errors.New(message)
    }

    return translation, nil
}
```

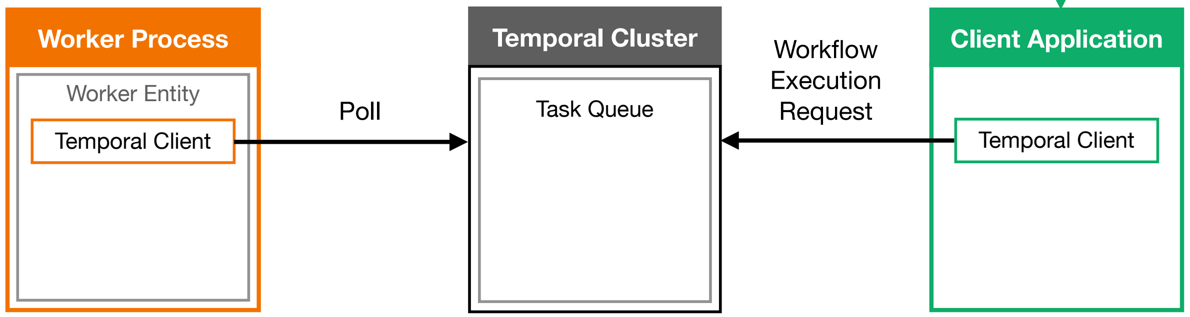
Workflow Definition

```go
package farewell

import (
    "time"

    "go.temporal.io/sdk/workflow"
)

func GreetSomeone(ctx workflow.Context, name string) (string, error) {
    options := workflow.ActivityOptions{
        StartToCloseTimeout: time.Second * 5,
    }
    ctx = workflow.WithActivityOptions(ctx, options)

    var spanishGreeting string
    err := workflow.ExecuteActivity(ctx, GreetInSpanish, name).Get(ctx, &spanishGreeting)
    if err != nil {
        return "", err
    }

    var spanishFarewell string
    err = workflow.ExecuteActivity(ctx, FarewellInSpanish, name).Get(ctx, &spanishFarewell)
    if err != nil {
        return "", err
    }

    var helloGoodbye = "\n" + spanishGreeting + "\n" + spanishFarewell

    return helloGoodbye, nil
}
```

Worker Initialization

```go
package main

import (
    "log"
    farewell "temporal101/exercises/farewell-workflow/solution"

    "go.temporal.io/sdk/client"
    "go.temporal.io/sdk/worker"
)

func main() {
    c, err := client.Dial(client.Options{})
    if err != nil {
        log.Fatalln("Unable to create client", err)
    }
    defer c.Close()

    w := worker.New(c, "greeting-tasks", worker.Options{})

    w.RegisterWorkflow(farewell.GreetSomeone)
    w.RegisterActivity(farewell.GreetInSpanish)
    w.RegisterActivity(farewell.FarewellInSpanish)

    err = w.Run(worker.InterruptCh())
    if err != nil {
        log.Fatalln("Unable to start worker", err)
    }
}
```

```go
// ... this is code within your own application (for example, a web application, mobile app, etc.)

 c, err := client.Dial(client.Options{})
 if err != nil {
     log.Fatalln("unable to create Temporal client", err)
 }
 defer c.Close()

options := client.StartWorkflowOptions{
    ID:         "greeting-workflow",
    TaskQueue: "greeting-tasks",
}

we, err := c.ExecuteWorkflow(context.Background(), options, farewell.GreetSomeone, os.Args[1])
if err != nil {
    log.Fatalln("Unable to execute workflow", err)
}
log.Println("Started workflow", "WorkflowID", we.GetID(), "RunID", we.GetRunID())

var result string
err = we.Get(context.Background(), &result)
if err != nil {
    log.Fatalln("Unable get workflow result", err)
}
log.Println("Workflow result:", result)

 // ... other application-specific code might follow
```



Activity Definitions

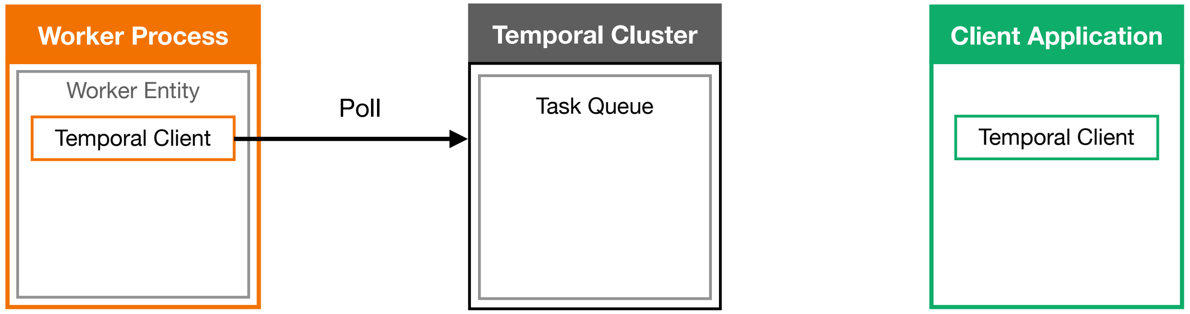
```go
package farewell    // import statements omitted for brevity

func GreetInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-greeting", name)
    return greeting, err
}

func FarewellInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-farewell", name)
    return greeting, err
}

// utility function for making calls to the microservices
func callService(stem string, name string) (string, error) {
    base := "http://localhost:9999/" + stem + "?name=%s"
    url := fmt.Sprintf(base, url.QueryEscape(name))

    resp, err := http.Get(url)
    if err != nil {
        return "", err
    }
    defer resp.Body.Close()

    body, err := ioutil.ReadAll(resp.Body)
    if err != nil {
        return "", err
    }

    translation := string(body)
    status := resp.StatusCode
    if status >= 400 {
        message := fmt.Sprintf("HTTP Error %d: %s", status, translation)
        return "", errors.New(message)
    }

    return translation, nil
}
```

Workflow Definition

```go
package farewell

import (
    "time"

    "go.temporal.io/sdk/workflow"
)

func GreetSomeone(ctx workflow.Context, name string) (string, error) {
    options := workflow.ActivityOptions{
        StartToCloseTimeout: time.Second * 5,
    }
    ctx = workflow.WithActivityOptions(ctx, options)

    var spanishGreeting string
    err := workflow.ExecuteActivity(ctx, GreetInSpanish, name).Get(ctx, &spanishGreeting)
    if err != nil {
        return "", err
    }

    var spanishFarewell string
    err = workflow.ExecuteActivity(ctx, FarewellInSpanish, name).Get(ctx, &spanishFarewell)
    if err != nil {
        return "", err
    }

    var helloGoodbye = "\n" + spanishGreeting + "\n" + spanishFarewell

    return helloGoodbye, nil
}
```

Worker Initialization

```go
package main

import (
    "log"
    farewell "temporal101/exercises/farewell-workflow/solution"

    "go.temporal.io/sdk/client"
    "go.temporal.io/sdk/worker"
)

func main() {
    c, err := client.Dial(client.Options{})
    if err != nil {
        log.Fatalln("Unable to create client", err)
    }
    defer c.Close()

    w := worker.New(c, "greeting-tasks", worker.Options{})

    w.RegisterWorkflow(farewell.GreetSomeone)
    w.RegisterActivity(farewell.GreetInSpanish)
    w.RegisterActivity(farewell.FarewellInSpanish)

    err = w.Run(worker.InterruptCh())
    if err != nil {
        log.Fatalln("Unable to start worker", err)
    }
}
```



Event History

WorkflowExecutionStarted

Activity Definitions

```
package farewell    // import statements omitted for brevity

func GreetInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-greeting", name)
    return greeting, err
}

func FarewellInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-farewell", name)
    return greeting, err
}

// utility function for making calls to the microservices
func callService(stem string, name string) (string, error) {
    base := "http://localhost:9999/" + stem + "?name=%s"
    url := fmt.Sprintf(base, url.QueryEscape(name))

    resp, err := http.Get(url)
    if err != nil {
        return "", err
    }
    defer resp.Body.Close()

    body, err := ioutil.ReadAll(resp.Body)
    if err != nil {
        return "", err
    }

    translation := string(body)
    status := resp.StatusCode
    if status >= 400 {
        message := fmt.Sprintf("HTTP Error %d: %s", status, translation)
        return "", errors.New(message)
    }

    return translation, nil
}
```

Workflow Definition

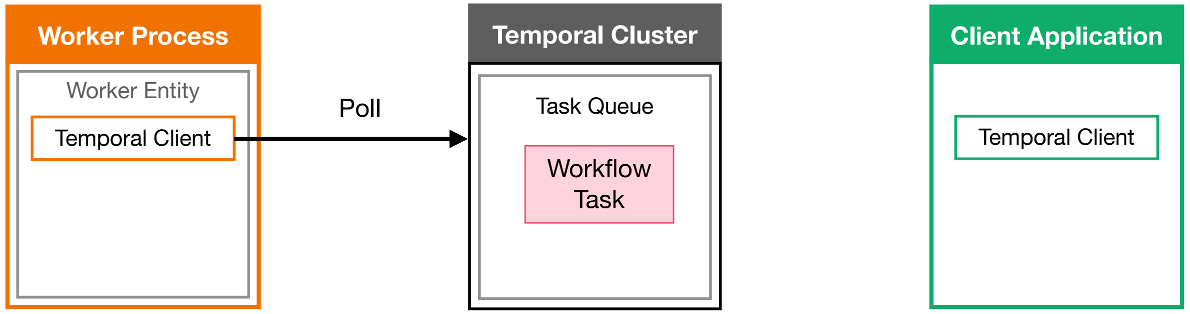
```
package farewell

import (
    "time"

    "go.temporal.io/sdk/workflow"
)

func GreetSomeone(ctx workflow.Context, name string) (string, error) {
    options := workflow.ActivityOptions{
        StartToCloseTimeout: time.Second * 5,
    }
    ctx = workflow.WithActivityOptions(ctx, options)

    var spanishGreeting string
    err := workflow.ExecuteActivity(ctx, GreetInSpanish, name).Get(ctx, &spanishGreeting)
    if err != nil {
        return "", err
    }

    var spanishFarewell string
    err = workflow.ExecuteActivity(ctx, FarewellInSpanish, name).Get(ctx, &spanishFarewell)
    if err != nil {
        return "", err
    }

    var helloGoodbye = "\n" + spanishGreeting + "\n" + spanishFarewell

    return helloGoodbye, nil
}
```

Worker Initialization

```
package main

import (
    "log"
    farewell "temporal101/exercises/farewell-workflow/solution"

    "go.temporal.io/sdk/client"
    "go.temporal.io/sdk/worker"
)

func main() {
    c, err := client.Dial(client.Options{})
    if err != nil {
        log.Fatalln("Unable to create client", err)
    }
    defer c.Close()

    w := worker.New(c, "greeting-tasks", worker.Options{})

    w.RegisterWorkflow(farewell.GreetSomeone)
    w.RegisterActivity(farewell.GreetInSpanish)
    w.RegisterActivity(farewell.FarewellInSpanish)

    err = w.Run(worker.InterruptCh())
    if err != nil {
        log.Fatalln("Unable to start worker", err)
    }
}
```

Event History

```
WorkflowExecutionStarted
WorkflowTaskScheduled
```



Activity Definitions

```go
package farewell    // import statements omitted for brevity

func GreetInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-greeting", name)
    return greeting, err
}

func FarewellInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-farewell", name)
    return greeting, err
}

// utility function for making calls to the microservices
func callService(stem string, name string) (string, error) {
    base := "http://localhost:9999/" + stem + "?name=%s"
    url := fmt.Sprintf(base, url.QueryEscape(name))

    resp, err := http.Get(url)
    if err != nil {
        return "", err
    }
    defer resp.Body.Close()

    body, err := ioutil.ReadAll(resp.Body)
    if err != nil {
        return "", err
    }

    translation := string(body)
    status := resp.StatusCode
    if status >= 400 {
        message := fmt.Sprintf("HTTP Error %d: %s", status, translation)
        return "", errors.New(message)
    }

    return translation, nil
}
```

Workflow Definition

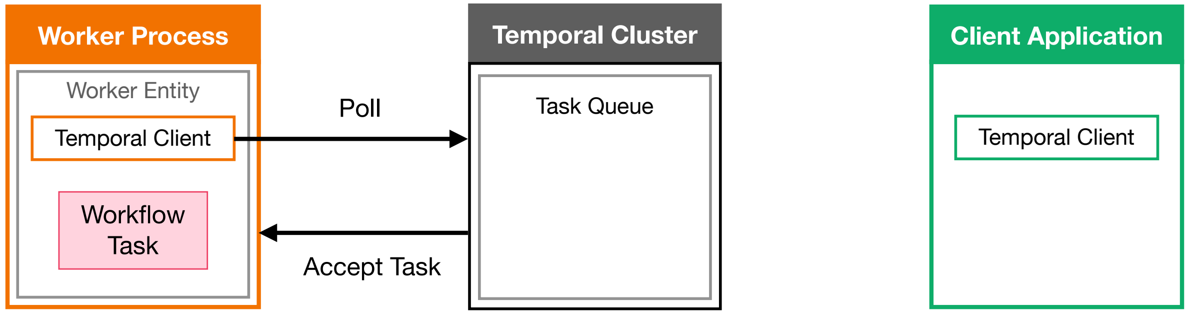
```go
package farewell

import (
    "time"

    "go.temporal.io/sdk/workflow"
)

func GreetSomeone(ctx workflow.Context, name string) (string, error) {
    options := workflow.ActivityOptions{
        StartToCloseTimeout: time.Second * 5,
    }
    ctx = workflow.WithActivityOptions(ctx, options)

    var spanishGreeting string
    err := workflow.ExecuteActivity(ctx, GreetInSpanish, name).Get(ctx, &spanishGreeting)
    if err != nil {
        return "", err
    }

    var spanishFarewell string
    err = workflow.ExecuteActivity(ctx, FarewellInSpanish, name).Get(ctx, &spanishFarewell)
    if err != nil {
        return "", err
    }

    var helloGoodbye = "\n" + spanishGreeting + "\n" + spanishFarewell

    return helloGoodbye, nil
}
```

Worker Initialization

```go
package main

import (
    "log"
    farewell "temporal101/exercises/farewell-workflow/solution"

    "go.temporal.io/sdk/client"
    "go.temporal.io/sdk/worker"
)

func main() {
    c, err := client.Dial(client.Options{})
    if err != nil {
        log.Fatalln("Unable to create client", err)
    }
    defer c.Close()

    w := worker.New(c, "greeting-tasks", worker.Options{})

    w.RegisterWorkflow(farewell.GreetSomeone)
    w.RegisterActivity(farewell.GreetInSpanish)
    w.RegisterActivity(farewell.FarewellInSpanish)

    err = w.Run(worker.InterruptCh())
    if err != nil {
        log.Fatalln("Unable to start worker", err)
    }
}
```

Event History

```
WorkflowExecutionStarted
WorkflowTaskScheduled
WorkflowTaskStarted
```



Activity Definitions

```
package farewell    // import statements omitted for brevity

func GreetInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-greeting", name)
    return greeting, err
}

func FarewellInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-farewell", name)
    return greeting, err
}

// utility function for making calls to the microservices
func callService(stem string, name string) (string, error) {
    base := "http://localhost:9999/" + stem + "?name=%s"
    url := fmt.Sprintf(base, url.QueryEscape(name))

    resp, err := http.Get(url)
    if err != nil {
        return "", err
    }
    defer resp.Body.Close()

    body, err := ioutil.ReadAll(resp.Body)
    if err != nil {
        return "", err
    }

    translation := string(body)
    status := resp.StatusCode
    if status >= 400 {
        message := fmt.Sprintf("HTTP Error %d: %s", status, translation)
        return "", errors.New(message)
    }

    return translation, nil
}
```

Workflow Definition

```
package farewell

import (
    "time"

    "go.temporal.io/sdk/workflow"
)

func GreetSomeone(ctx workflow.Context, name string) (string, error) {
    options := workflow.ActivityOptions{
        StartToCloseTimeout: time.Second * 5,
    }
    ctx = workflow.WithActivityOptions(ctx, options)

    var spanishGreeting string
    err := workflow.ExecuteActivity(ctx, GreetInSpanish, name).Get(ctx, &spanishGreeting)
    if err != nil {
        return "", err
    }

    var spanishFarewell string
    err = workflow.ExecuteActivity(ctx, FarewellInSpanish, name).Get(ctx, &spanishFarewell)
    if err != nil {
        return "", err
    }

    var helloGoodbye = "\n" + spanishGreeting + "\n" + spanishFarewell

    return helloGoodbye, nil
}
```

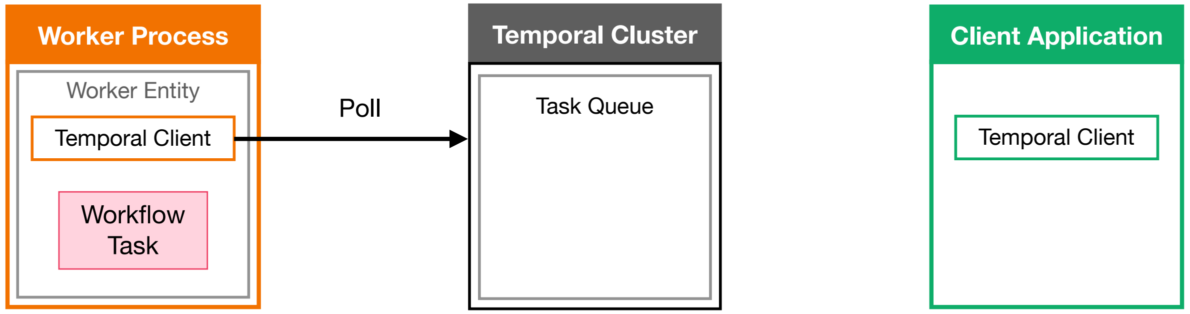
Worker Initialization

```
package main

import (
    "log"
    farewell "temporal101/exercises/farewell-workflow/solution"

    "go.temporal.io/sdk/client"
    "go.temporal.io/sdk/worker"
)

func main() {
    c, err := client.Dial(client.Options{})
    if err != nil {
        log.Fatalln("Unable to create client", err)
    }
    defer c.Close()

    w := worker.New(c, "greeting-tasks", worker.Options{})

    w.RegisterWorkflow(farewell.GreetSomeone)
    w.RegisterActivity(farewell.GreetInSpanish)
    w.RegisterActivity(farewell.FarewellInSpanish)

    err = w.Run(worker.InterruptCh())
    if err != nil {
        log.Fatalln("Unable to start worker", err)
    }
}
```

```
// ... code above has been omitted from this excerpt

func GreetSomeone(ctx workflow.Context, name string) (string, error) {
    options := workflow.ActivityOptions{
        StartToCloseTimeout: time.Second * 5,
    }
    ctx = workflow.WithActivityOptions(ctx, options)

    var spanishGreeting string
    err := workflow.ExecuteActivity(ctx, GreetInSpanish, name).Get(ctx, &spanishGreeting)
    if err != nil {
        return "", err
    }

    var spanishFarewell string
    err = workflow.ExecuteActivity(ctx, FarewellInSpanish, name).Get(ctx, &spanishFarewell)
    if err != nil {
        return "", err
    }

    var helloGoodbye = "\n" + spanishGreeting + "\n" + spanishFarewell

    return helloGoodbye, nil
}
```

Event History

```
WorkflowExecutionStarted
WorkflowTaskScheduled
WorkflowTaskStarted
```



Activity Definitions

```
package farewell    // import statements omitted for brevity

func GreetInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-greeting", name)
    return greeting, err
}

func FarewellInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-farewell", name)
    return greeting, err
}

// utility function for making calls to the microservices
func callService(stem string, name string) (string, error) {
    base := "http://localhost:9999/" + stem + "?name=%s"
    url := fmt.Sprintf(base, url.QueryEscape(name))

    resp, err := http.Get(url)
    if err != nil {
        return "", err
    }
    defer resp.Body.Close()

    body, err := ioutil.ReadAll(resp.Body)
    if err != nil {
        return "", err
    }

    translation := string(body)
    status := resp.StatusCode
    if status >= 400 {
        message := fmt.Sprintf("HTTP Error %d: %s", status, translation)
        return "", errors.New(message)
    }

    return translation, nil
}
```

Workflow Definition

```
package farewell

import (
    "time"

    "go.temporal.io/sdk/workflow"
)

func GreetSomeone(ctx workflow.Context, name string) (string, error) {
    options := workflow.ActivityOptions{
        StartToCloseTimeout: time.Second * 5,
    }
    ctx = workflow.WithActivityOptions(ctx, options)

    var spanishGreeting string
    err := workflow.ExecuteActivity(ctx, GreetInSpanish, name).Get(ctx, &spanishGreeting)
    if err != nil {
        return "", err
    }

    var spanishFarewell string
    err = workflow.ExecuteActivity(ctx, FarewellInSpanish, name).Get(ctx, &spanishFarewell)
    if err != nil {
        return "", err
    }

    var helloGoodbye = "\n" + spanishGreeting + "\n" + spanishFarewell

    return helloGoodbye, nil
}
```

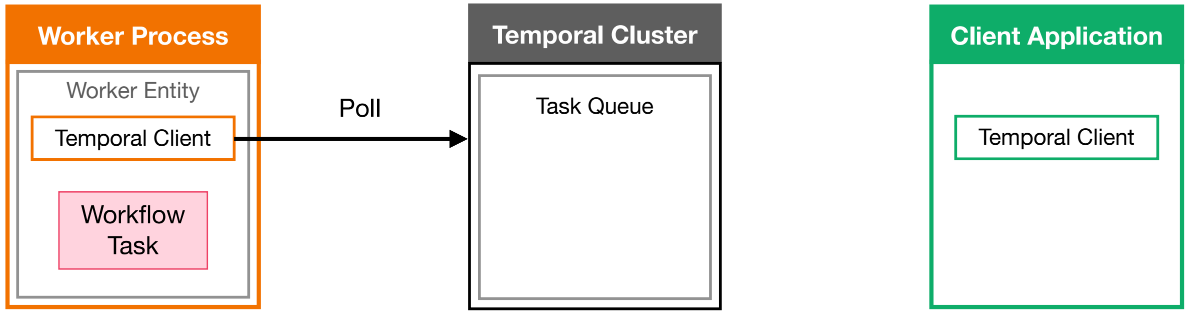
Worker Initialization

```
package main

import (
    "log"
    farewell "temporal101/exercises/farewell-workflow/solution"

    "go.temporal.io/sdk/client"
    "go.temporal.io/sdk/worker"
)

func main() {
    c, err := client.Dial(client.Options{})
    if err != nil {
        log.Fatalln("Unable to create client", err)
    }
    defer c.Close()

    w := worker.New(c, "greeting-tasks", worker.Options{})

    w.RegisterWorkflow(farewell.GreetSomeone)
    w.RegisterActivity(farewell.GreetInSpanish)
    w.RegisterActivity(farewell.FarewellInSpanish)

    err = w.Run(worker.InterruptCh())
    if err != nil {
        log.Fatalln("Unable to start worker", err)
    }
}
```

```
// ... code above has been omitted from this excerpt


func GreetSomeone(ctx workflow.Context, name string) (string, error) {
    options := workflow.ActivityOptions{
        StartToCloseTimeout: time.Second * 5,
    }
    ctx = workflow.WithActivityOptions(ctx, options)

    var spanishGreeting string
    err := workflow.ExecuteActivity(ctx, GreetInSpanish, name).Get(ctx, &spanishGreeting)
    if err != nil {
        return "", err
    }

    var spanishFarewell string
    err = workflow.ExecuteActivity(ctx, FarewellInSpanish, name).Get(ctx, &spanishFarewell)
    if err != nil {
        return "", err
    }

    var helloGoodbye = "\n" + spanishGreeting + "\n" + spanishFarewell

    return helloGoodbye, nil
}
```

Event History

WorkflowExecutionStarted
WorkflowTaskScheduled
WorkflowTaskStarted



Activity Definitions

```go
package farewell    // import statements omitted for brevity

func GreetInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-greeting", name)
    return greeting, err
}

func FarewellInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-farewell", name)
    return greeting, err
}

// utility function for making calls to the microservices
func callService(stem string, name string) (string, error) {
    base := "http://localhost:9999/" + stem + "?name=%s"
    url := fmt.Sprintf(base, url.QueryEscape(name))

    resp, err := http.Get(url)
    if err != nil {
        return "", err
    }
    defer resp.Body.Close()

    body, err := ioutil.ReadAll(resp.Body)
    if err != nil {
        return "", err
    }

    translation := string(body)
    status := resp.StatusCode
    if status >= 400 {
        message := fmt.Sprintf("HTTP Error %d: %s", status, translation)
        return "", errors.New(message)
    }

    return translation, nil
}
```

Workflow Definition

```go
package farewell

import (
    "time"

    "go.temporal.io/sdk/workflow"
)

func GreetSomeone(ctx workflow.Context, name string) (string, error) {
    options := workflow.ActivityOptions{
        StartToCloseTimeout: time.Second * 5,
    }
    ctx = workflow.WithActivityOptions(ctx, options)

    var spanishGreeting string
    err := workflow.ExecuteActivity(ctx, GreetInSpanish, name).Get(ctx, &spanishGreeting)
    if err != nil {
        return "", err
    }

    var spanishFarewell string
    err = workflow.ExecuteActivity(ctx, FarewellInSpanish, name).Get(ctx, &spanishFarewell)
    if err != nil {
        return "", err
    }

    var helloGoodbye = "\n" + spanishGreeting + "\n" + spanishFarewell

    return helloGoodbye, nil
}
```

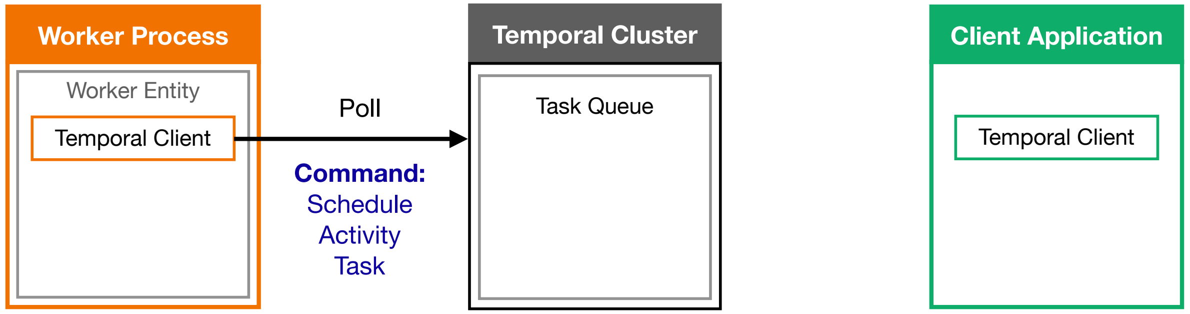
Worker Initialization

```go
package main

import (
    "log"
    farewell "temporal101/exercises/farewell-workflow/solution"

    "go.temporal.io/sdk/client"
    "go.temporal.io/sdk/worker"
)

func main() {
    c, err := client.Dial(client.Options{})
    if err != nil {
        log.Fatalln("Unable to create client", err)
    }
    defer c.Close()

    w := worker.New(c, "greeting-tasks", worker.Options{})

    w.RegisterWorkflow(farewell.GreetSomeone)
    w.RegisterActivity(farewell.GreetInSpanish)
    w.RegisterActivity(farewell.FarewellInSpanish)

    err = w.Run(worker.InterruptCh())
    if err != nil {
        log.Fatalln("Unable to start worker", err)
    }
}
```

```go
// ... code above has been omitted from this excerpt


func GreetSomeone(ctx workflow.Context, name string) (string, error) {
    options := workflow.ActivityOptions{
        StartToCloseTimeout: time.Second * 5,
    }
    ctx = workflow.WithActivityOptions(ctx, options)

    var spanishGreeting string
    err := workflow.ExecuteActivity(ctx, GreetInSpanish, name).Get(ctx, &spanishGreeting)
    if err != nil {
        return "", err
    }

    var spanishFarewell string
    err = workflow.ExecuteActivity(ctx, FarewellInSpanish, name).Get(ctx, &spanishFarewell)
    if err != nil {
        return "", err
    }

    var helloGoodbye = "\n" + spanishGreeting + "\n" + spanishFarewell

    return helloGoodbye, nil
}
```

Event History

WorkflowExecutionStarted
WorkflowTaskScheduled
WorkflowTaskStarted
WorkflowTaskCompleted



Activity Definitions

```
package farewell    // import statements omitted for brevity

func GreetInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-greeting", name)
    return greeting, err
}

func FarewellInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-farewell", name)
    return greeting, err
}

// utility function for making calls to the microservices
func callService(stem string, name string) (string, error) {
    base := "http://localhost:9999/" + stem + "?name=%s"
    url := fmt.Sprintf(base, url.QueryEscape(name))

    resp, err := http.Get(url)
    if err != nil {
        return "", err
    }
    defer resp.Body.Close()

    body, err := ioutil.ReadAll(resp.Body)
    if err != nil {
        return "", err
    }

    translation := string(body)
    status := resp.StatusCode
    if status >= 400 {
        message := fmt.Sprintf("HTTP Error %d: %s", status, translation)
        return "", errors.New(message)
    }

    return translation, nil
}
```

Workflow Definition

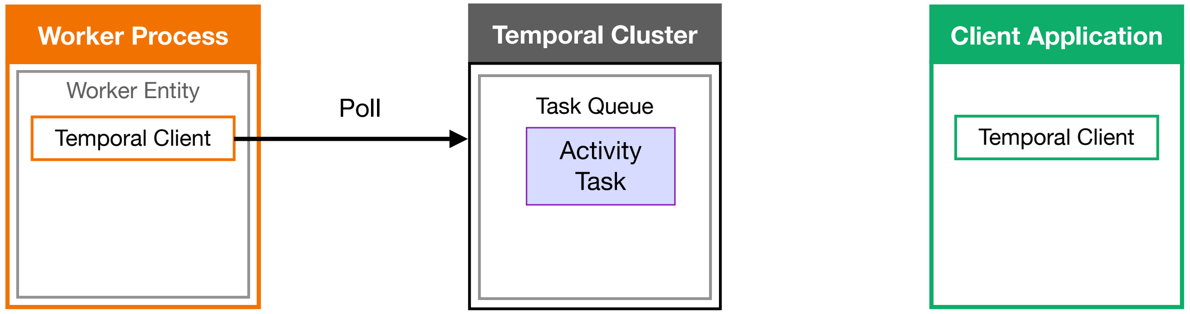
```
package farewell

import (
    "time"

    "go.temporal.io/sdk/workflow"
)

func GreetSomeone(ctx workflow.Context, name string) (string, error) {
    options := workflow.ActivityOptions{
        StartToCloseTimeout: time.Second * 5,
    }
    ctx = workflow.WithActivityOptions(ctx, options)

    var spanishGreeting string
    err := workflow.ExecuteActivity(ctx, GreetInSpanish, name).Get(ctx, &spanishGreeting)
    if err != nil {
        return "", err
    }

    var spanishFarewell string
    err = workflow.ExecuteActivity(ctx, FarewellInSpanish, name).Get(ctx, &spanishFarewell)
    if err != nil {
        return "", err
    }

    var helloGoodbye = "\n" + spanishGreeting + "\n" + spanishFarewell

    return helloGoodbye, nil
}
```

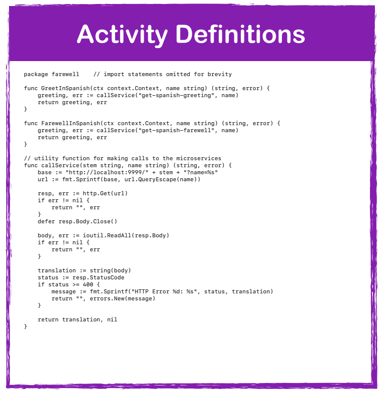
Worker Initialization

```
package main

import (
    "log"
    farewell "temporal101/exercises/farewell-workflow/solution"

    "go.temporal.io/sdk/client"
    "go.temporal.io/sdk/worker"
)

func main() {
    c, err := client.Dial(client.Options{})
    if err != nil {
        log.Fatalln("Unable to create client", err)
    }
    defer c.Close()

    w := worker.New(c, "greeting-tasks", worker.Options{})

    w.RegisterWorkflow(farewell.GreetSomeone)
    w.RegisterActivity(farewell.GreetInSpanish)
    w.RegisterActivity(farewell.FarewellInSpanish)

    err = w.Run(worker.InterruptCh())
    if err != nil {
        log.Fatalln("Unable to start worker", err)
    }
}
```

Event History

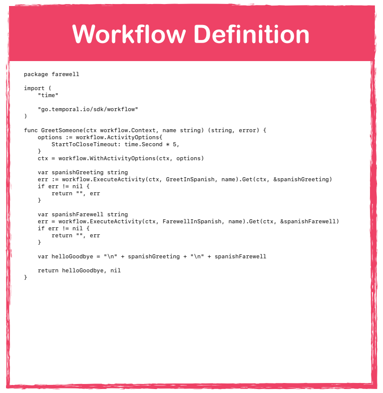
WorkflowExecutionStarted	
WorkflowTaskScheduled	
WorkflowTaskStarted	
WorkflowTaskCompleted	
ActivityTaskScheduled	(Greeting)





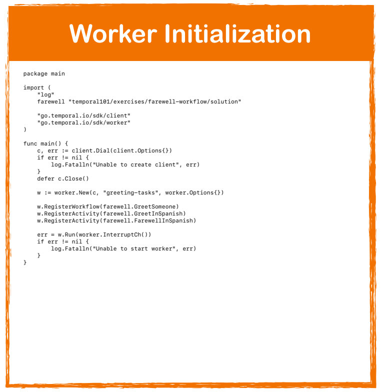
Activity Definitions

```
package farewell    // import statements omitted for brevity

func GreetInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-greeting", name)
    return greeting, err
}

func FarewellInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-farewell", name)
    return greeting, err
}

// utility function for making calls to the microservices
func callService(stem string, name string) (string, error) {
    base := "http://localhost:9999/" + stem + "?name=%s"
    url := fmt.Sprintf(base, url.QueryEscape(name))

    resp, err := http.Get(url)
    if err != nil {
        return "", err
    }
    defer resp.Body.Close()

    body, err := ioutil.ReadAll(resp.Body)
    if err != nil {
        return "", err
    }

    translation := string(body)
    status := resp.StatusCode
    if status >= 400 {
        message := fmt.Sprintf("HTTP Error %d: %s", status, translation)
        return "", errors.New(message)
    }

    return translation, nil
}
```



Workflow Definition

```
package farewell

import (
    "time"

    "go.temporal.io/sdk/workflow"
)

func GreetSomeone(ctx workflow.Context, name string) (string, error) {
    options := workflow.ActivityOptions{
        StartToCloseTimeout: time.Second * 5,
    }
    ctx = workflow.WithActivityOptions(ctx, options)

    var spanishGreeting string
    err := workflow.ExecuteActivity(ctx, GreetInSpanish, name).Get(ctx, &spanishGreeting)
    if err != nil {
        return "", err
    }

    var spanishFarewell string
    err = workflow.ExecuteActivity(ctx, FarewellInSpanish, name).Get(ctx, &spanishFarewell)
    if err != nil {
        return "", err
    }

    var helloGoodbye = "\n" + spanishGreeting + "\n" + spanishFarewell

    return helloGoodbye, nil
}
```

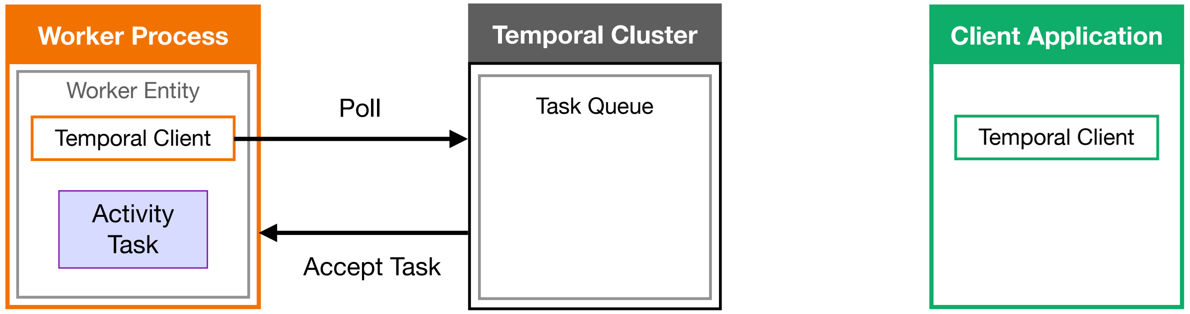


Worker Initialization

```
package main

import (
    "log"
    farewell "temporal101/exercises/farewell-workflow/solution"

    "go.temporal.io/sdk/client"
    "go.temporal.io/sdk/worker"
)

func main() {
    c, err := client.Dial(client.Options{})
    if err != nil {
        log.Fatalln("Unable to create client", err)
    }
    defer c.Close()

    w := worker.New(c, "greeting-tasks", worker.Options{})

    w.RegisterWorkflow(farewell.GreetSomeone)
    w.RegisterActivity(farewell.GreetInSpanish)
    w.RegisterActivity(farewell.FarewellInSpanish)

    err = w.Run(worker.InterruptCh())
    if err != nil {
        log.Fatalln("Unable to start worker", err)
    }
}
```

Event History

WorkflowExecutionStarted	
WorkflowTaskScheduled	
WorkflowTaskStarted	
WorkflowTaskCompleted	
ActivityTaskScheduled	(Greeting)
ActivityTaskStarted	(Greeting)



Activity Definitions

```
package farewell    // import statements omitted for brevity

func GreetInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-greeting", name)
    return greeting, err
}

func FarewellInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-farewell", name)
    return greeting, err
}

// utility function for making calls to the microservices
func callService(stem string, name string) (string, error) {
    base := "http://localhost:9999/" + stem + "?name=%s"
    url := fmt.Sprintf(base, url.QueryEscape(name))

    resp, err := http.Get(url)
    if err != nil {
        return "", err
    }
    defer resp.Body.Close()

    body, err := ioutil.ReadAll(resp.Body)
    if err != nil {
        return "", err
    }

    translation := string(body)
    status := resp.StatusCode
    if status >= 400 {
        message := fmt.Sprintf("HTTP Error %d: %s", status, translation)
        return "", errors.New(message)
    }

    return translation, nil
}
```

Workflow Definition

```
package farewell

import (
    "time"

    "go.temporal.io/sdk/workflow"
)

func GreetSomeone(ctx workflow.Context, name string) (string, error) {
    options := workflow.ActivityOptions{
        StartToCloseTimeout: time.Second * 5,
    }
    ctx = workflow.WithActivityOptions(ctx, options)

    var spanishGreeting string
    err := workflow.ExecuteActivity(ctx, GreetInSpanish, name).Get(ctx, &spanishGreeting)
    if err != nil {
        return "", err
    }

    var spanishFarewell string
    err = workflow.ExecuteActivity(ctx, FarewellInSpanish, name).Get(ctx, &spanishFarewell)
    if err != nil {
        return "", err
    }

    var helloGoodbye = "\n" + spanishGreeting + "\n" + spanishFarewell

    return helloGoodbye, nil
}
```

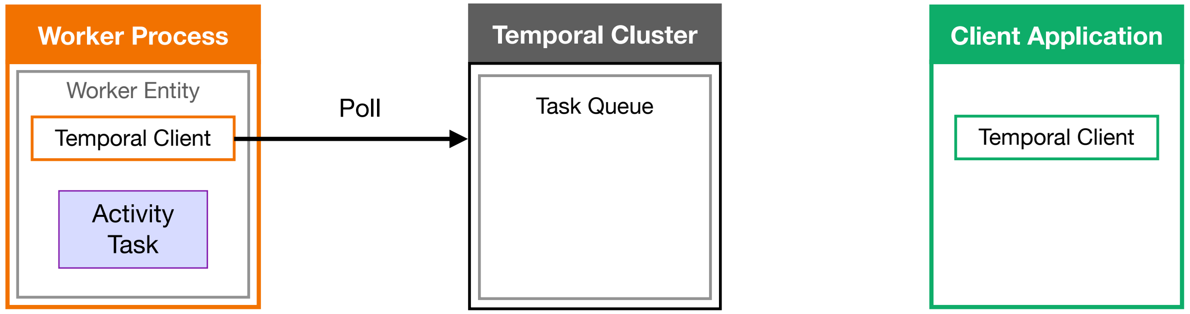
Worker Initialization

```
package main

import (
    "log"
    farewell "temporal101/exercises/farewell-workflow/solution"

    "go.temporal.io/sdk/client"
    "go.temporal.io/sdk/worker"
)

func main() {
    c, err := client.Dial(client.Options{})
    if err != nil {
        log.Fatalln("Unable to create client", err)
    }
    defer c.Close()

    w := worker.New(c, "greeting-tasks", worker.Options{})

    w.RegisterWorkflow(farewell.GreetSomeone)
    w.RegisterActivity(farewell.GreetInSpanish)
    w.RegisterActivity(farewell.FarewellInSpanish)

    err = w.Run(worker.InterruptCh())
    if err != nil {
        log.Fatalln("Unable to start worker", err)
    }
}
```

```
// import statements and unused code omitted from this example
func GreetInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-greeting", name)
    return greeting, err
}

// utility function for making calls to the microservices
func callService(stem string, name string) (string, error) {
    base := "http://localhost:9999/" + stem + "?name=%s"
    url := fmt.Sprintf(base, url.QueryEscape(name))

    resp, err := http.Get(url)
    if err != nil {
        return "", err
    }
    defer resp.Body.Close()

    body, err := ioutil.ReadAll(resp.Body)
    if err != nil {
        return "", err
    }

    translation := string(body)
    status := resp.StatusCode
    if status >= 400 {
        message := fmt.Sprintf("HTTP Error %d: %s", status, translation)
        return "", errors.New(message)
    }

    return translation, nil
}
```

Event History

WorkflowExecutionStarted	
WorkflowTaskScheduled	
WorkflowTaskStarted	
WorkflowTaskCompleted	
ActivityTaskScheduled	(Greeting)
ActivityTaskStarted	(Greeting)



Activity Definitions

```
package farewell    // import statements omitted for brevity

func GreetInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-greeting", name)
    return greeting, err
}

func FarewellInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-farewell", name)
    return greeting, err
}

// utility function for making calls to the microservices
func callService(stem string, name string) (string, error) {
    base := "http://localhost:9999/" + stem + "?name=%s"
    url := fmt.Sprintf(base, url.QueryEscape(name))

    resp, err := http.Get(url)
    if err != nil {
        return "", err
    }
    defer resp.Body.Close()

    body, err := ioutil.ReadAll(resp.Body)
    if err != nil {
        return "", err
    }

    translation := string(body)
    status := resp.StatusCode
    if status >= 400 {
        message := fmt.Sprintf("HTTP Error %d: %s", status, translation)
        return "", errors.New(message)
    }

    return translation, nil
}
```

Workflow Definition

```
package farewell

import (
    "time"

    "go.temporal.io/sdk/workflow"
)

func GreetSomeone(ctx workflow.Context, name string) (string, error) {
    options := workflow.ActivityOptions{
        StartToCloseTimeout: time.Second * 5,
    }
    ctx = workflow.WithActivityOptions(ctx, options)

    var spanishGreeting string
    err := workflow.ExecuteActivity(ctx, GreetInSpanish, name).Get(ctx, &spanishGreeting)
    if err != nil {
        return "", err
    }

    var spanishFarewell string
    err = workflow.ExecuteActivity(ctx, FarewellInSpanish, name).Get(ctx, &spanishFarewell)
    if err != nil {
        return "", err
    }

    var helloGoodbye = "\n" + spanishGreeting + "\n" + spanishFarewell

    return helloGoodbye, nil
}
```

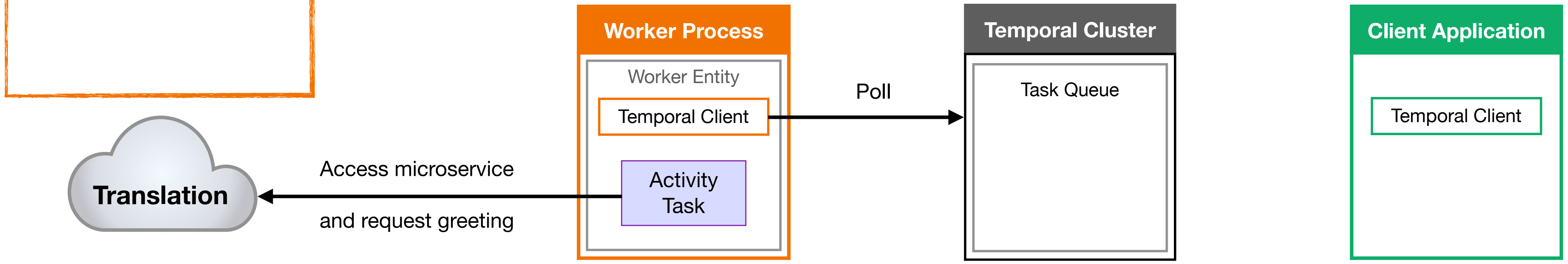
Worker Initialization

```
package main

import (
    "log"
    farewell "temporal101/exercises/farewell-workflow/solution"

    "go.temporal.io/sdk/client"
    "go.temporal.io/sdk/worker"
)

func main() {
    c, err := client.Dial(client.Options{})
    if err != nil {
        log.Fatalln("Unable to create client", err)
    }
    defer c.Close()

    w := worker.New(c, "greeting-tasks", worker.Options{})

    w.RegisterWorkflow(farewell.GreetSomeone)
    w.RegisterActivity(farewell.GreetInSpanish)
    w.RegisterActivity(farewell.FarewellInSpanish)

    err = w.Run(worker.InterruptCh())
    if err != nil {
        log.Fatalln("Unable to start worker", err)
    }
}
```

```
// import statements and unused code omitted from this example
func GreetInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-greeting", name)
    return greeting, err
}

// utility function for making calls to the microservices
func callService(stem string, name string) (string, error) {
    base := "http://localhost:9999/" + stem + "?name=%s"
    url := fmt.Sprintf(base, url.QueryEscape(name))

    resp, err := http.Get(url)
    if err != nil {
        return "", err
    }
    defer resp.Body.Close()

    body, err := ioutil.ReadAll(resp.Body)
    if err != nil {
        return "", err
    }

    translation := string(body)
    status := resp.StatusCode
    if status >= 400 {
        message := fmt.Sprintf("HTTP Error %d: %s", status, translation)
        return "", errors.New(message)
    }

    return translation, nil
}
```

Event History

WorkflowExecutionStarted	
WorkflowTaskScheduled	
WorkflowTaskStarted	
WorkflowTaskCompleted	
ActivityTaskScheduled	(Greeting)
ActivityTaskStarted	(Greeting)



Activity Definitions

```go
package farewell    // import statements omitted for brevity

func GreetInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-greeting", name)
    return greeting, err
}

func FarewellInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-farewell", name)
    return greeting, err
}

// utility function for making calls to the microservices
func callService(stem string, name string) (string, error) {
    base := "http://localhost:9999/" + stem + "?name=%s"
    url := fmt.Sprintf(base, url.QueryEscape(name))

    resp, err := http.Get(url)
    if err != nil {
        return "", err
    }
    defer resp.Body.Close()

    body, err := ioutil.ReadAll(resp.Body)
    if err != nil {
        return "", err
    }

    translation := string(body)
    status := resp.StatusCode
    if status >= 400 {
        message := fmt.Sprintf("HTTP Error %d: %s", status, translation)
        return "", errors.New(message)
    }

    return translation, nil
}
```

Workflow Definition

```go
package farewell

import (
    "time"

    "go.temporal.io/sdk/workflow"
)

func GreetSomeone(ctx workflow.Context, name string) (string, error) {
    options := workflow.ActivityOptions{
        StartToCloseTimeout: time.Second * 5,
    }
    ctx = workflow.WithActivityOptions(ctx, options)

    var spanishGreeting string
    err := workflow.ExecuteActivity(ctx, GreetInSpanish, name).Get(ctx, &spanishGreeting)
    if err != nil {
        return "", err
    }

    var spanishFarewell string
    err = workflow.ExecuteActivity(ctx, FarewellInSpanish, name).Get(ctx, &spanishFarewell)
    if err != nil {
        return "", err
    }

    var helloGoodbye = "\n" + spanishGreeting + "\n" + spanishFarewell

    return helloGoodbye, nil
}
```

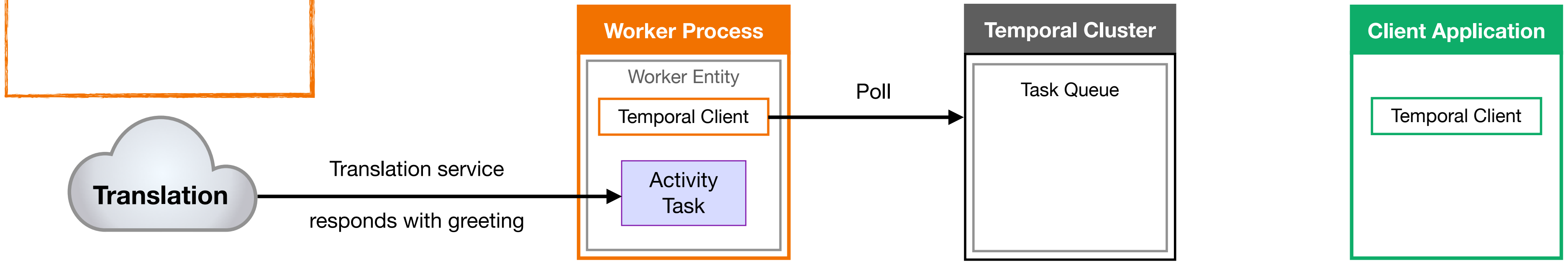
Worker Initialization

```go
package main

import (
    "log"
    farewell "temporal101/exercises/farewell-workflow/solution"

    "go.temporal.io/sdk/client"
    "go.temporal.io/sdk/worker"
)

func main() {
    c, err := client.Dial(client.Options{})
    if err != nil {
        log.Fatalln("Unable to create client", err)
    }
    defer c.Close()

    w := worker.New(c, "greeting-tasks", worker.Options{})

    w.RegisterWorkflow(farewell.GreetSomeone)
    w.RegisterActivity(farewell.GreetInSpanish)
    w.RegisterActivity(farewell.FarewellInSpanish)

    err = w.Run(worker.InterruptCh())
    if err != nil {
        log.Fatalln("Unable to start worker", err)
    }
}
```

```go
// import statements and unused code omitted from this example
func GreetInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-greeting", name)
    return greeting, err
}

// utility function for making calls to the microservices
func callService(stem string, name string) (string, error) {
    base := "http://localhost:9999/" + stem + "?name=%s"
    url := fmt.Sprintf(base, url.QueryEscape(name))

    resp, err := http.Get(url)
    if err != nil {
        return "", err
    }
    defer resp.Body.Close()

    body, err := ioutil.ReadAll(resp.Body)
    if err != nil {
        return "", err
    }

    translation := string(body)
    status := resp.StatusCode
    if status >= 400 {
        message := fmt.Sprintf("HTTP Error %d: %s", status, translation)
        return "", errors.New(message)
    }

    return translation, nil
}
```

Event History

WorkflowExecutionStarted	
WorkflowTaskScheduled	
WorkflowTaskStarted	
WorkflowTaskCompleted	
ActivityTaskScheduled	(Greeting)
ActivityTaskStarted	(Greeting)



Activity Definitions

```go
package farewell    // import statements omitted for brevity

func GreetInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-greeting", name)
    return greeting, err
}

func FarewellInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-farewell", name)
    return greeting, err
}

// utility function for making calls to the microservices
func callService(stem string, name string) (string, error) {
    base := "http://localhost:9999/" + stem + "?name=%s"
    url := fmt.Sprintf(base, url.QueryEscape(name))

    resp, err := http.Get(url)
    if err != nil {
        return "", err
    }
    defer resp.Body.Close()

    body, err := ioutil.ReadAll(resp.Body)
    if err != nil {
        return "", err
    }

    translation := string(body)
    status := resp.StatusCode
    if status >= 400 {
        message := fmt.Sprintf("HTTP Error %d: %s", status, translation)
        return "", errors.New(message)
    }

    return translation, nil
}
```

Workflow Definition

```go
package farewell

import (
    "time"

    "go.temporal.io/sdk/workflow"
)

func GreetSomeone(ctx workflow.Context, name string) (string, error) {
    options := workflow.ActivityOptions{
        StartToCloseTimeout: time.Second * 5,
    }
    ctx = workflow.WithActivityOptions(ctx, options)

    var spanishGreeting string
    err := workflow.ExecuteActivity(ctx, GreetInSpanish, name).Get(ctx, &spanishGreeting)
    if err != nil {
        return "", err
    }

    var spanishFarewell string
    err = workflow.ExecuteActivity(ctx, FarewellInSpanish, name).Get(ctx, &spanishFarewell)
    if err != nil {
        return "", err
    }

    var helloGoodbye = "\n" + spanishGreeting + "\n" + spanishFarewell

    return helloGoodbye, nil
}
```

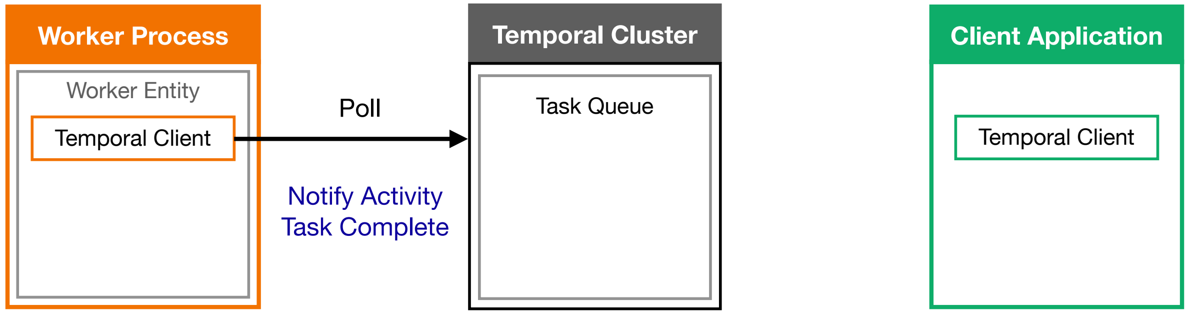
Worker Initialization

```go
package main

import (
    "log"
    farewell "temporal101/exercises/farewell-workflow/solution"

    "go.temporal.io/sdk/client"
    "go.temporal.io/sdk/worker"
)

func main() {
    c, err := client.Dial(client.Options{})
    if err != nil {
        log.Fatalln("Unable to create client", err)
    }
    defer c.Close()

    w := worker.New(c, "greeting-tasks", worker.Options{})

    w.RegisterWorkflow(farewell.GreetSomeone)
    w.RegisterActivity(farewell.GreetInSpanish)
    w.RegisterActivity(farewell.FarewellInSpanish)

    err = w.Run(worker.InterruptCh())
    if err != nil {
        log.Fatalln("Unable to start worker", err)
    }
}
```

```go
// import statements and unused code omitted from this example
func GreetInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-greeting", name)
    return greeting, err
}

// utility function for making calls to the microservices
func callService(stem string, name string) (string, error) {
    base := "http://localhost:9999/" + stem + "?name=%s"
    url := fmt.Sprintf(base, url.QueryEscape(name))

    resp, err := http.Get(url)
    if err != nil {
        return "", err
    }
    defer resp.Body.Close()

    body, err := ioutil.ReadAll(resp.Body)
    if err != nil {
        return "", err
    }

    translation := string(body)
    status := resp.StatusCode
    if status >= 400 {
        message := fmt.Sprintf("HTTP Error %d: %s", status, translation)
        return "", errors.New(message)
    }

    return translation, nil
}
```

Event History

WorkflowExecutionStarted	
WorkflowTaskScheduled	
WorkflowTaskStarted	
WorkflowTaskCompleted	
ActivityTaskScheduled	(Greeting)
ActivityTaskStarted	(Greeting)
ActivityTaskCompleted	(Greeting)



Activity Definitions

```
package farewell    // import statements omitted for brevity

func GreetInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-greeting", name)
    return greeting, err
}

func FarewellInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-farewell", name)
    return greeting, err
}

// utility function for making calls to the microservices
func callService(stem string, name string) (string, error) {
    base := "http://localhost:9999/" + stem + "?name=%s"
    url := fmt.Sprintf(base, url.QueryEscape(name))

    resp, err := http.Get(url)
    if err != nil {
        return "", err
    }
    defer resp.Body.Close()

    body, err := ioutil.ReadAll(resp.Body)
    if err != nil {
        return "", err
    }

    translation := string(body)
    status := resp.StatusCode
    if status >= 400 {
        message := fmt.Sprintf("HTTP Error %d: %s", status, translation)
        return "", errors.New(message)
    }

    return translation, nil
}
```

Workflow Definition

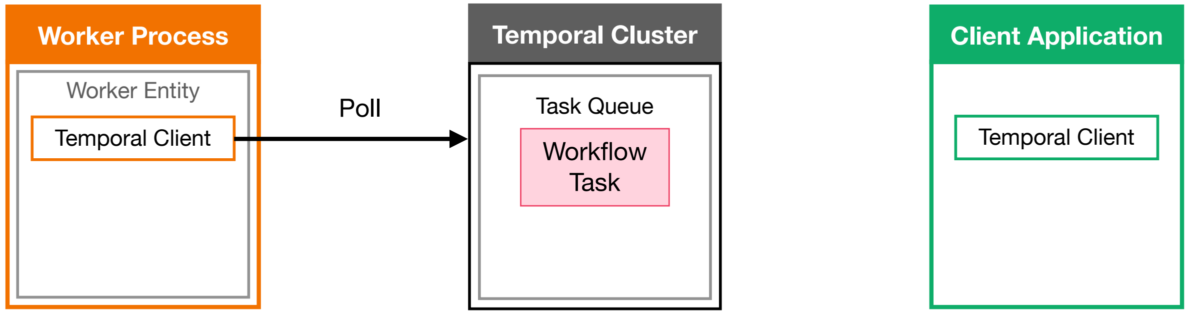
```
package farewell

import (
    "time"

    "go.temporal.io/sdk/workflow"
)

func GreetSomeone(ctx workflow.Context, name string) (string, error) {
    options := workflow.ActivityOptions{
        StartToCloseTimeout: time.Second * 5,
    }
    ctx = workflow.WithActivityOptions(ctx, options)

    var spanishGreeting string
    err := workflow.ExecuteActivity(ctx, GreetInSpanish, name).Get(ctx, &spanishGreeting)
    if err != nil {
        return "", err
    }

    var spanishFarewell string
    err = workflow.ExecuteActivity(ctx, FarewellInSpanish, name).Get(ctx, &spanishFarewell)
    if err != nil {
        return "", err
    }

    var helloGoodbye = "\n" + spanishGreeting + "\n" + spanishFarewell

    return helloGoodbye, nil
}
```

Worker Initialization

```
package main

import (
    "log"
    farewell "temporal101/exercises/farewell-workflow/solution"

    "go.temporal.io/sdk/client"
    "go.temporal.io/sdk/worker"
)

func main() {
    c, err := client.Dial(client.Options{})
    if err != nil {
        log.Fatalln("Unable to create client", err)
    }
    defer c.Close()

    w := worker.New(c, "greeting-tasks", worker.Options{})

    w.RegisterWorkflow(farewell.GreetSomeone)
    w.RegisterActivity(farewell.GreetInSpanish)
    w.RegisterActivity(farewell.FarewellInSpanish)

    err = w.Run(worker.InterruptCh())
    if err != nil {
        log.Fatalln("Unable to start worker", err)
    }
}
```

Event History

WorkflowExecutionStarted	
WorkflowTaskScheduled	
WorkflowTaskStarted	
WorkflowTaskCompleted	
ActivityTaskScheduled	(Greeting)
ActivityTaskStarted	(Greeting)
ActivityTaskCompleted	(Greeting)
WorkflowTaskScheduled	



Activity Definitions

```
package farewell    // import statements omitted for brevity

func GreetInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-greeting", name)
    return greeting, err
}

func FarewellInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-farewell", name)
    return greeting, err
}

// utility function for making calls to the microservices
func callService(stem string, name string) (string, error) {
    base := "http://localhost:9999/" + stem + "?name=%s"
    url := fmt.Sprintf(base, url.QueryEscape(name))

    resp, err := http.Get(url)
    if err != nil {
        return "", err
    }
    defer resp.Body.Close()

    body, err := ioutil.ReadAll(resp.Body)
    if err != nil {
        return "", err
    }

    translation := string(body)
    status := resp.StatusCode
    if status >= 400 {
        message := fmt.Sprintf("HTTP Error %d: %s", status, translation)
        return "", errors.New(message)
    }

    return translation, nil
}
```

Workflow Definition

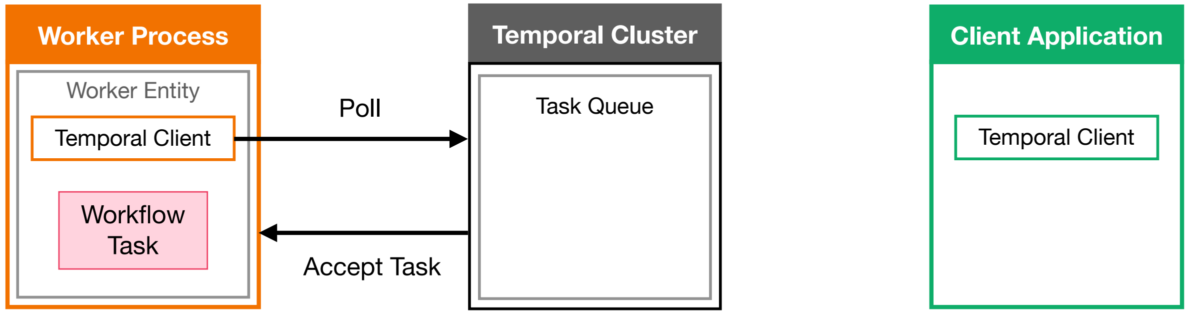
```
package farewell

import (
    "time"

    "go.temporal.io/sdk/workflow"
)

func GreetSomeone(ctx workflow.Context, name string) (string, error) {
    options := workflow.ActivityOptions{
        StartToCloseTimeout: time.Second * 5,
    }
    ctx = workflow.WithActivityOptions(ctx, options)

    var spanishGreeting string
    err := workflow.ExecuteActivity(ctx, GreetInSpanish, name).Get(ctx, &spanishGreeting)
    if err != nil {
        return "", err
    }

    var spanishFarewell string
    err = workflow.ExecuteActivity(ctx, FarewellInSpanish, name).Get(ctx, &spanishFarewell)
    if err != nil {
        return "", err
    }

    var helloGoodbye = "\n" + spanishGreeting + "\n" + spanishFarewell

    return helloGoodbye, nil
}
```

Worker Initialization

```
package main

import (
    "log"
    farewell "temporal101/exercises/farewell-workflow/solution"

    "go.temporal.io/sdk/client"
    "go.temporal.io/sdk/worker"
)

func main() {
    c, err := client.Dial(client.Options{})
    if err != nil {
        log.Fatalln("Unable to create client", err)
    }
    defer c.Close()

    w := worker.New(c, "greeting-tasks", worker.Options{})

    w.RegisterWorkflow(farewell.GreetSomeone)
    w.RegisterActivity(farewell.GreetInSpanish)
    w.RegisterActivity(farewell.FarewellInSpanish)

    err = w.Run(worker.InterruptCh())
    if err != nil {
        log.Fatalln("Unable to start worker", err)
    }
}
```

Event History

WorkflowExecutionStarted	
WorkflowTaskScheduled	
WorkflowTaskStarted	
WorkflowTaskCompleted	
ActivityTaskScheduled	(Greeting)
ActivityTaskStarted	(Greeting)
ActivityTaskCompleted	(Greeting)
WorkflowTaskScheduled	
WorkflowTaskStarted	



Activity Definitions

```
package farewell    // import statements omitted for brevity

func GreetInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-greeting", name)
    return greeting, err
}

func FarewellInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-farewell", name)
    return greeting, err
}

// utility function for making calls to the microservices
func callService(stem string, name string) (string, error) {
    base := "http://localhost:9999/" + stem + "?name=%s"
    url := fmt.Sprintf(base, url.QueryEscape(name))

    resp, err := http.Get(url)
    if err != nil {
        return "", err
    }
    defer resp.Body.Close()

    body, err := ioutil.ReadAll(resp.Body)
    if err != nil {
        return "", err
    }

    translation := string(body)
    status := resp.StatusCode
    if status >= 400 {
        message := fmt.Sprintf("HTTP Error %d: %s", status, translation)
        return "", errors.New(message)
    }

    return translation, nil
}
```

Workflow Definition

```
package farewell

import (
    "time"

    "go.temporal.io/sdk/workflow"
)

func GreetSomeone(ctx workflow.Context, name string) (string, error) {
    options := workflow.ActivityOptions{
        StartToCloseTimeout: time.Second * 5,
    }
    ctx = workflow.WithActivityOptions(ctx, options)

    var spanishGreeting string
    err := workflow.ExecuteActivity(ctx, GreetInSpanish, name).Get(ctx, &spanishGreeting)
    if err != nil {
        return "", err
    }

    var spanishFarewell string
    err = workflow.ExecuteActivity(ctx, FarewellInSpanish, name).Get(ctx, &spanishFarewell)
    if err != nil {
        return "", err
    }

    var helloGoodbye = "\n" + spanishGreeting + "\n" + spanishFarewell

    return helloGoodbye, nil
}
```

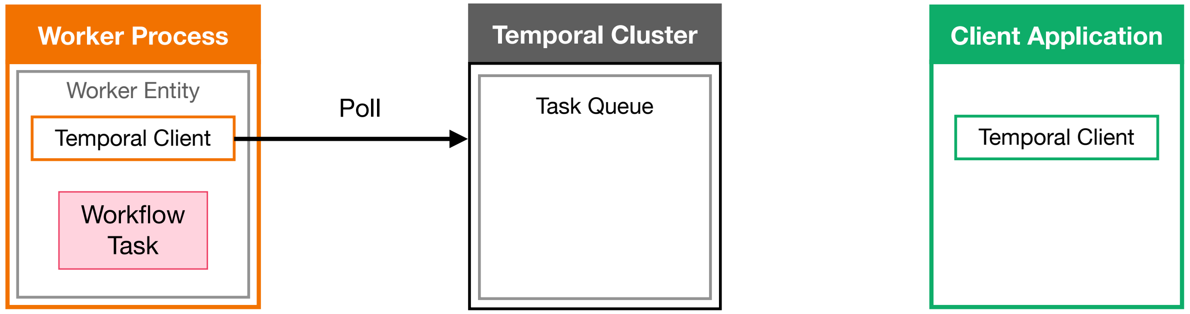
Worker Initialization

```
package main

import (
    "log"
    farewell "temporal101/exercises/farewell-workflow/solution"

    "go.temporal.io/sdk/client"
    "go.temporal.io/sdk/worker"
)

func main() {
    c, err := client.Dial(client.Options{})
    if err != nil {
        log.Fatalln("Unable to create client", err)
    }
    defer c.Close()

    w := worker.New(c, "greeting-tasks", worker.Options{})

    w.RegisterWorkflow(farewell.GreetSomeone)
    w.RegisterActivity(farewell.GreetInSpanish)
    w.RegisterActivity(farewell.FarewellInSpanish)

    err = w.Run(worker.InterruptCh())
    if err != nil {
        log.Fatalln("Unable to start worker", err)
    }
}
```

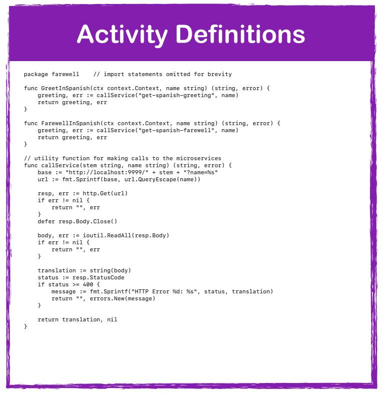
```
// ... code above has been omitted from this excerpt


func GreetSomeone(ctx workflow.Context, name string) (string, error) {
    options := workflow.ActivityOptions{
        StartToCloseTimeout: time.Second * 5,
    }
    ctx = workflow.WithActivityOptions(ctx, options)

    var spanishGreeting string
    err := workflow.ExecuteActivity(ctx, GreetInSpanish, name).Get(ctx, &spanishGreeting)
    if err != nil {
        return "", err
    }

    var spanishFarewell string
    err = workflow.ExecuteActivity(ctx, FarewellInSpanish, name).Get(ctx, &spanishFarewell)
    if err != nil {
        return "", err
    }

    var helloGoodbye = "\n" + spanishGreeting + "\n" + spanishFarewell

    return helloGoodbye, nil
}
```

Event History

WorkflowExecutionStarted	
WorkflowTaskScheduled	
WorkflowTaskStarted	
WorkflowTaskCompleted	
ActivityTaskScheduled	(Greeting)
ActivityTaskStarted	(Greeting)
ActivityTaskCompleted	(Greeting)
WorkflowTaskScheduled	
WorkflowTaskStarted	

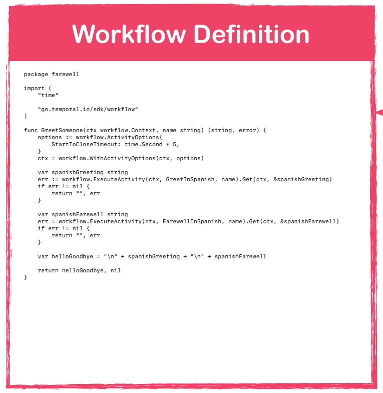


Activity Definitions



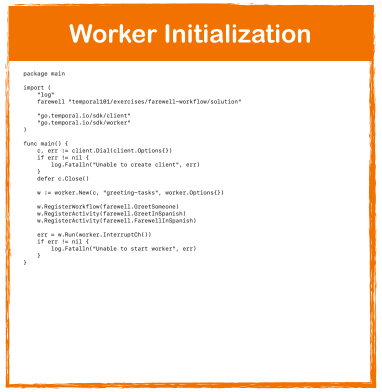
```
package farewell    // import statements omitted for brevity

func GreetInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-greeting", name)
    return greeting, err
}

func FarewellInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-farewell", name)
    return greeting, err
}

// utility function for making calls to the microservices
func callService(stem string, name string) (string, error) {
    base := "http://localhost:9999/" + stem + "?name=%s"
    url := fmt.Sprintf(base, url.QueryEscape(name))

    resp, err := http.Get(url)
    if err != nil {
        return "", err
    }
    defer resp.Body.Close()

    body, err := ioutil.ReadAll(resp.Body)
    if err != nil {
        return "", err
    }

    translation := string(body)
    status := resp.StatusCode
    if status >= 400 {
        message := fmt.Sprintf("HTTP Error %d: %s", status, translation)
        return "", errors.New(message)
    }

    return translation, nil
}
```

Workflow Definition



```
package farewell

import (
    "time"

    "go.temporal.io/sdk/workflow"
)

func GreetSomeone(ctx workflow.Context, name string) (string, error) {
    options := workflow.ActivityOptions{
        StartToCloseTimeout: time.Second * 5,
    }
    ctx = workflow.WithActivityOptions(ctx, options)

    var spanishGreeting string
    err := workflow.ExecuteActivity(ctx, GreetInSpanish, name).Get(ctx, &spanishGreeting)
    if err != nil {
        return "", err
    }

    var spanishFarewell string
    err = workflow.ExecuteActivity(ctx, FarewellInSpanish, name).Get(ctx, &spanishFarewell)
    if err != nil {
        return "", err
    }

    var helloGoodbye = "\n" + spanishGreeting + "\n" + spanishFarewell

    return helloGoodbye, nil
}
```

Worker Initialization

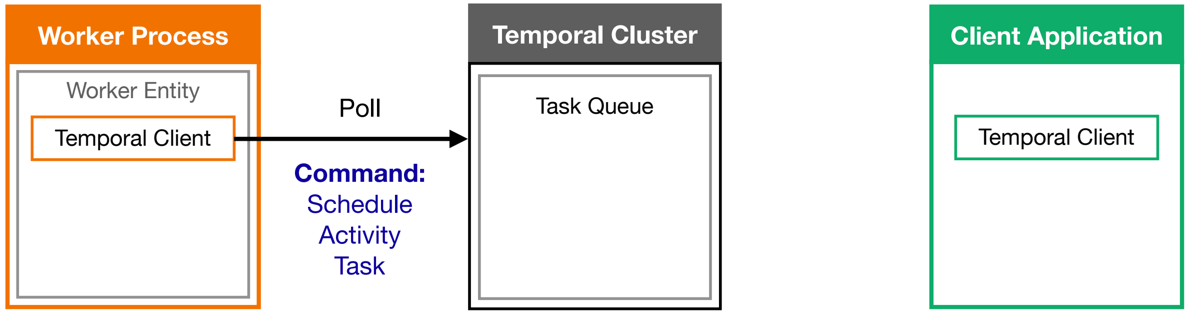


```
package main

import (
    "log"
    farewell "temporal101/exercises/farewell-workflow/solution"

    "go.temporal.io/sdk/client"
    "go.temporal.io/sdk/worker"
)

func main() {
    c, err := client.Dial(client.Options{})
    if err != nil {
        log.Fatalln("Unable to create client", err)
    }
    defer c.Close()

    w := worker.New(c, "greeting-tasks", worker.Options{})

    w.RegisterWorkflow(farewell.GreetSomeone)
    w.RegisterActivity(farewell.GreetInSpanish)
    w.RegisterActivity(farewell.FarewellInSpanish)

    err = w.Run(worker.InterruptCh())
    if err != nil {
        log.Fatalln("Unable to start worker", err)
    }
}
```

```
// ... code above has been omitted from this excerpt


func GreetSomeone(ctx workflow.Context, name string) (string, error) {
    options := workflow.ActivityOptions{
        StartToCloseTimeout: time.Second * 5,
    }
    ctx = workflow.WithActivityOptions(ctx, options)

    var spanishGreeting string
    err := workflow.ExecuteActivity(ctx, GreetInSpanish, name).Get(ctx, &spanishGreeting)
    if err != nil {
        return "", err
    }

    var spanishFarewell string
    err = workflow.ExecuteActivity(ctx, FarewellInSpanish, name).Get(ctx, &spanishFarewell)
    if err != nil {
        return "", err
    }

    var helloGoodbye = "\n" + spanishGreeting + "\n" + spanishFarewell

    return helloGoodbye, nil
}
```

Event History

WorkflowExecutionStarted	
WorkflowTaskScheduled	
WorkflowTaskStarted	
WorkflowTaskCompleted	
ActivityTaskScheduled	(Greeting)
ActivityTaskStarted	(Greeting)
ActivityTaskCompleted	(Greeting)
WorkflowTaskScheduled	
WorkflowTaskStarted	
WorkflowTaskCompleted	



Activity Definitions

```go
package farewell    // import statements omitted for brevity

func GreetInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-greeting", name)
    return greeting, err
}

func FarewellInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-farewell", name)
    return greeting, err
}

// utility function for making calls to the microservices
func callService(stem string, name string) (string, error) {
    base := "http://localhost:9999/" + stem + "?name=%s"
    url := fmt.Sprintf(base, url.QueryEscape(name))

    resp, err := http.Get(url)
    if err != nil {
        return "", err
    }
    defer resp.Body.Close()

    body, err := ioutil.ReadAll(resp.Body)
    if err != nil {
        return "", err
    }

    translation := string(body)
    status := resp.StatusCode
    if status >= 400 {
        message := fmt.Sprintf("HTTP Error %d: %s", status, translation)
        return "", errors.New(message)
    }

    return translation, nil
}
```

Workflow Definition

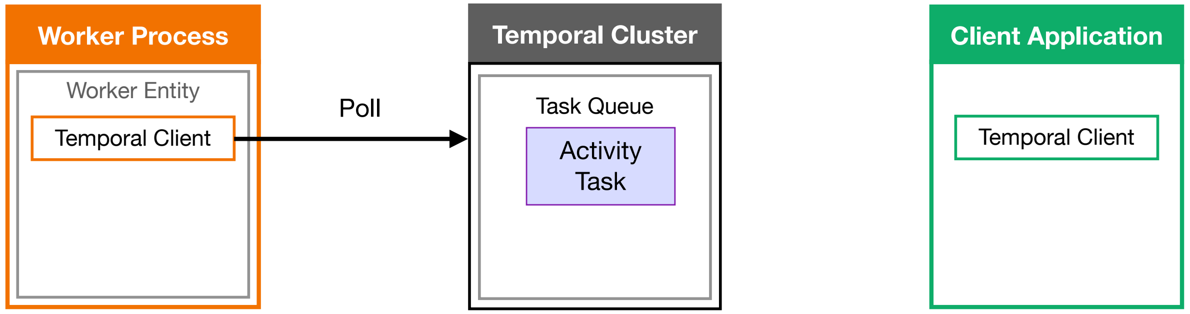
```go
package farewell

import (
    "time"

    "go.temporal.io/sdk/workflow"
)

func GreetSomeone(ctx workflow.Context, name string) (string, error) {
    options := workflow.ActivityOptions{
        StartToCloseTimeout: time.Second * 5,
    }
    ctx = workflow.WithActivityOptions(ctx, options)

    var spanishGreeting string
    err := workflow.ExecuteActivity(ctx, GreetInSpanish, name).Get(ctx, &spanishGreeting)
    if err != nil {
        return "", err
    }

    var spanishFarewell string
    err = workflow.ExecuteActivity(ctx, FarewellInSpanish, name).Get(ctx, &spanishFarewell)
    if err != nil {
        return "", err
    }

    var helloGoodbye = "\n" + spanishGreeting + "\n" + spanishFarewell

    return helloGoodbye, nil
}
```

Worker Initialization

```go
package main

import (
    "log"
    farewell "temporal101/exercises/farewell-workflow/solution"

    "go.temporal.io/sdk/client"
    "go.temporal.io/sdk/worker"
)

func main() {
    c, err := client.Dial(client.Options{})
    if err != nil {
        log.Fatalln("Unable to create client", err)
    }
    defer c.Close()

    w := worker.New(c, "greeting-tasks", worker.Options{})

    w.RegisterWorkflow(farewell.GreetSomeone)
    w.RegisterActivity(farewell.GreetInSpanish)
    w.RegisterActivity(farewell.FarewellInSpanish)

    err = w.Run(worker.InterruptCh())
    if err != nil {
        log.Fatalln("Unable to start worker", err)
    }
}
```

Event History

WorkflowExecutionStarted	
WorkflowTaskScheduled	
WorkflowTaskStarted	
WorkflowTaskCompleted	
ActivityTaskScheduled	(Greeting)
ActivityTaskStarted	(Greeting)
ActivityTaskCompleted	(Greeting)
WorkflowTaskScheduled	
WorkflowTaskStarted	
WorkflowTaskCompleted	
ActivityTaskScheduled	(Farewell)



Activity Definitions

```
package farewell    // import statements omitted for brevity

func GreetInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-greeting", name)
    return greeting, err
}

func FarewellInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-farewell", name)
    return greeting, err
}

// utility function for making calls to the microservices
func callService(stem string, name string) (string, error) {
    base := "http://localhost:9999/" + stem + "?name=%s"
    url := fmt.Sprintf(base, url.QueryEscape(name))

    resp, err := http.Get(url)
    if err != nil {
        return "", err
    }
    defer resp.Body.Close()

    body, err := ioutil.ReadAll(resp.Body)
    if err != nil {
        return "", err
    }

    translation := string(body)
    status := resp.StatusCode
    if status >= 400 {
        message := fmt.Sprintf("HTTP Error %d: %s", status, translation)
        return "", errors.New(message)
    }

    return translation, nil
}
```

Workflow Definition

```
package farewell

import (
    "time"

    "go.temporal.io/sdk/workflow"
)

func GreetSomeone(ctx workflow.Context, name string) (string, error) {
    options := workflow.ActivityOptions{
        StartToCloseTimeout: time.Second * 5,
    }
    ctx = workflow.WithActivityOptions(ctx, options)

    var spanishGreeting string
    err := workflow.ExecuteActivity(ctx, GreetInSpanish, name).Get(ctx, &spanishGreeting)
    if err != nil {
        return "", err
    }

    var spanishFarewell string
    err = workflow.ExecuteActivity(ctx, FarewellInSpanish, name).Get(ctx, &spanishFarewell)
    if err != nil {
        return "", err
    }

    var helloGoodbye = "\n" + spanishGreeting + "\n" + spanishFarewell

    return helloGoodbye, nil
}
```

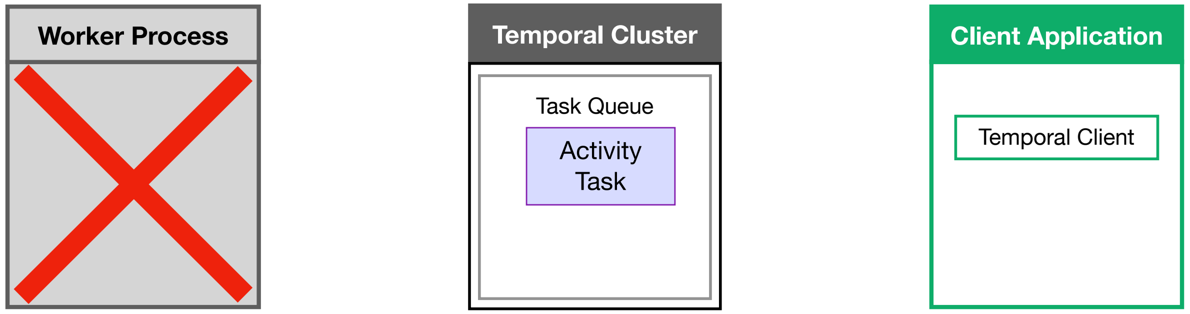
Worker Initialization

```
package main

import (
    "log"
    farewell "temporal101/exercises/farewell-workflow/solution"

    "go.temporal.io/sdk/client"
    "go.temporal.io/sdk/worker"
)

func main() {
    c, err := client.Dial(client.Options{})
    if err != nil {
        log.Fatalln("Unable to create client", err)
    }
    defer c.Close()

    w := worker.New(c, "greeting-tasks", worker.Options{})

    w.RegisterWorkflow(farewell.GreetSomeone)
    w.RegisterActivity(farewell.GreetInSpanish)
    w.RegisterActivity(farewell.FarewellInSpanish)

    err = w.Run(worker.InterruptCh())
    if err != nil {
        log.Fatalln("Unable to start worker", err)
    }
}
```

What happens if the Worker crashes?

Event History

WorkflowExecutionStarted	
WorkflowTaskScheduled	
WorkflowTaskStarted	
WorkflowTaskCompleted	
ActivityTaskScheduled	(Greeting)
ActivityTaskStarted	(Greeting)
ActivityTaskCompleted	(Greeting)
WorkflowTaskScheduled	
WorkflowTaskStarted	
WorkflowTaskCompleted	
ActivityTaskScheduled	(Farewell)



Worker Process

Temporal Cluster

Task Queue

Activity Task

Client Application

Temporal Client

Activity Definitions

```go
package farewell    // import statements omitted for brevity

func GreetInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-greeting", name)
    return greeting, err
}

func FarewellInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-farewell", name)
    return greeting, err
}

// utility function for making calls to the microservices
func callService(stem string, name string) (string, error) {
    base := "http://localhost:9999/" + stem + "?name=%s"
    url := fmt.Sprintf(base, url.QueryEscape(name))

    resp, err := http.Get(url)
    if err != nil {
        return "", err
    }
    defer resp.Body.Close()

    body, err := ioutil.ReadAll(resp.Body)
    if err != nil {
        return "", err
    }

    translation := string(body)
    status := resp.StatusCode
    if status >= 400 {
        message := fmt.Sprintf("HTTP Error %d: %s", status, translation)
        return "", errors.New(message)
    }

    return translation, nil
}
```

Workflow Definition

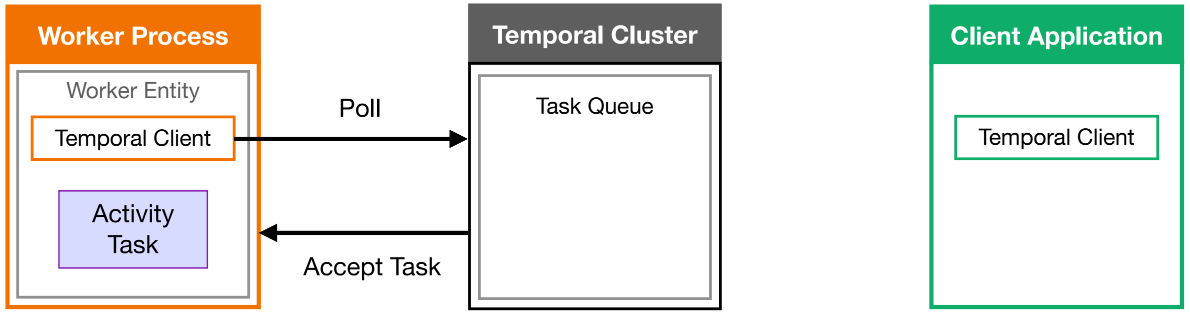
```go
package farewell

import (
    "time"

    "go.temporal.io/sdk/workflow"
)

func GreetSomeone(ctx workflow.Context, name string) (string, error) {
    options := workflow.ActivityOptions{
        StartToCloseTimeout: time.Second * 5,
    }
    ctx = workflow.WithActivityOptions(ctx, options)

    var spanishGreeting string
    err := workflow.ExecuteActivity(ctx, GreetInSpanish, name).Get(ctx, &spanishGreeting)
    if err != nil {
        return "", err
    }

    var spanishFarewell string
    err = workflow.ExecuteActivity(ctx, FarewellInSpanish, name).Get(ctx, &spanishFarewell)
    if err != nil {
        return "", err
    }

    var helloGoodbye = "\n" + spanishGreeting + "\n" + spanishFarewell

    return helloGoodbye, nil
}
```

Worker Initialization

```go
package main

import (
    "log"
    farewell "temporal101/exercises/farewell-workflow/solution"

    "go.temporal.io/sdk/client"
    "go.temporal.io/sdk/worker"
)

func main() {
    c, err := client.Dial(client.Options{})
    if err != nil {
        log.Fatalln("Unable to create client", err)
    }
    defer c.Close()

    w := worker.New(c, "greeting-tasks", worker.Options{})

    w.RegisterWorkflow(farewell.GreetSomeone)
    w.RegisterActivity(farewell.GreetInSpanish)
    w.RegisterActivity(farewell.FarewellInSpanish)

    err = w.Run(worker.InterruptCh())
    if err != nil {
        log.Fatalln("Unable to start worker", err)
    }
}
```

Event History

WorkflowExecutionStarted	
WorkflowTaskScheduled	
WorkflowTaskStarted	
WorkflowTaskCompleted	
ActivityTaskScheduled	(Greeting)
ActivityTaskStarted	(Greeting)
ActivityTaskCompleted	(Greeting)
WorkflowTaskScheduled	
WorkflowTaskStarted	
WorkflowTaskCompleted	
ActivityTaskScheduled	(Farewell)
ActivityTaskStarted	(Farewell)



Activity Definitions

```go
package farewell    // import statements omitted for brevity

func GreetInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-greeting", name)
    return greeting, err
}

func FarewellInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-farewell", name)
    return greeting, err
}

// utility function for making calls to the microservices
func callService(stem string, name string) (string, error) {
    base := "http://localhost:9999/" + stem + "?name=%s"
    url := fmt.Sprintf(base, url.QueryEscape(name))

    resp, err := http.Get(url)
    if err != nil {
        return "", err
    }
    defer resp.Body.Close()

    body, err := ioutil.ReadAll(resp.Body)
    if err != nil {
        return "", err
    }

    translation := string(body)
    status := resp.StatusCode
    if status >= 400 {
        message := fmt.Sprintf("HTTP Error %d: %s", status, translation)
        return "", errors.New(message)
    }

    return translation, nil
}
```

Workflow Definition

```go
package farewell

import (
    "time"

    "go.temporal.io/sdk/workflow"
)

func GreetSomeone(ctx workflow.Context, name string) (string, error) {
    options := workflow.ActivityOptions{
        StartToCloseTimeout: time.Second * 5,
    }
    ctx = workflow.WithActivityOptions(ctx, options)

    var spanishGreeting string
    err := workflow.ExecuteActivity(ctx, GreetInSpanish, name).Get(ctx, &spanishGreeting)
    if err != nil {
        return "", err
    }

    var spanishFarewell string
    err = workflow.ExecuteActivity(ctx, FarewellInSpanish, name).Get(ctx, &spanishFarewell)
    if err != nil {
        return "", err
    }

    var helloGoodbye = "\n" + spanishGreeting + "\n" + spanishFarewell

    return helloGoodbye, nil
}
```

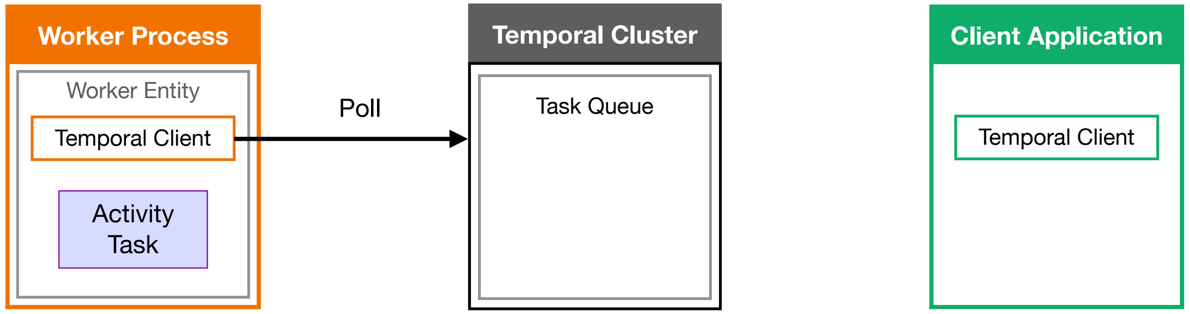
Worker Initialization

```go
package main

import (
    "log"
    farewell "temporal101/exercises/farewell-workflow/solution"

    "go.temporal.io/sdk/client"
    "go.temporal.io/sdk/worker"
)

func main() {
    c, err := client.Dial(client.Options{})
    if err != nil {
        log.Fatalln("Unable to create client", err)
    }
    defer c.Close()

    w := worker.New(c, "greeting-tasks", worker.Options{})

    w.RegisterWorkflow(farewell.GreetSomeone)
    w.RegisterActivity(farewell.GreetInSpanish)
    w.RegisterActivity(farewell.FarewellInSpanish)

    err = w.Run(worker.InterruptCh())
    if err != nil {
        log.Fatalln("Unable to start worker", err)
    }
}
```

```go
// import statements and unused code omitted from this example
func FarewellInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-greeting", name)
    return greeting, err
}

// utility function for making calls to the microservices
func callService(stem string, name string) (string, error) {
    base := "http://localhost:9999/" + stem + "?name=%s"
    url := fmt.Sprintf(base, url.QueryEscape(name))

    resp, err := http.Get(url)
    if err != nil {
        return "", err
    }
    defer resp.Body.Close()

    body, err := ioutil.ReadAll(resp.Body)
    if err != nil {
        return "", err
    }

    translation := string(body)
    status := resp.StatusCode
    if status >= 400 {
        message := fmt.Sprintf("HTTP Error %d: %s", status, translation)
        return "", errors.New(message)
    }

    return translation, nil
}
```

Event History

WorkflowExecutionStarted	
WorkflowTaskScheduled	
WorkflowTaskStarted	
WorkflowTaskCompleted	
ActivityTaskScheduled	(Greeting)
ActivityTaskStarted	(Greeting)
ActivityTaskCompleted	(Greeting)
WorkflowTaskScheduled	
WorkflowTaskStarted	
WorkflowTaskCompleted	
ActivityTaskScheduled	(Farewell)
ActivityTaskStarted	(Farewell)



Activity Definitions

```
package farewell    // import statements omitted for brevity

func GreetInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-greeting", name)
    return greeting, err
}

func FarewellInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-farewell", name)
    return greeting, err
}

// utility function for making calls to the microservices
func callService(stem string, name string) (string, error) {
    base := "http://localhost:9999/" + stem + "?name=%s"
    url := fmt.Sprintf(base, url.QueryEscape(name))

    resp, err := http.Get(url)
    if err != nil {
        return "", err
    }
    defer resp.Body.Close()

    body, err := ioutil.ReadAll(resp.Body)
    if err != nil {
        return "", err
    }

    translation := string(body)
    status := resp.StatusCode
    if status >= 400 {
        message := fmt.Sprintf("HTTP Error %d: %s", status, translation)
        return "", errors.New(message)
    }

    return translation, nil
}
```

Workflow Definition

```
package farewell

import (
    "time"

    "go.temporal.io/sdk/workflow"
)

func GreetSomeone(ctx workflow.Context, name string) (string, error) {
    options := workflow.ActivityOptions{
        StartToCloseTimeout: time.Second * 5,
    }
    ctx = workflow.WithActivityOptions(ctx, options)

    var spanishGreeting string
    err := workflow.ExecuteActivity(ctx, GreetInSpanish, name).Get(ctx, &spanishGreeting)
    if err != nil {
        return "", err
    }

    var spanishFarewell string
    err = workflow.ExecuteActivity(ctx, FarewellInSpanish, name).Get(ctx, &spanishFarewell)
    if err != nil {
        return "", err
    }

    var helloGoodbye = "\n" + spanishGreeting + "\n" + spanishFarewell

    return helloGoodbye, nil
}
```

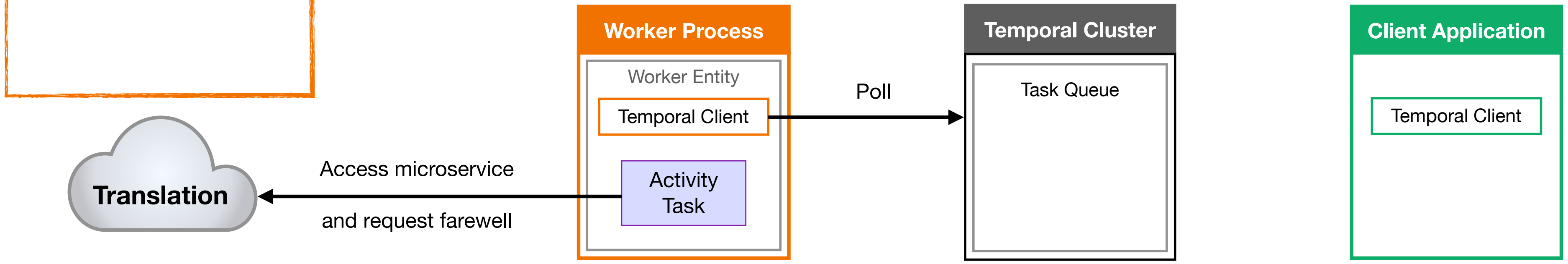
Worker Initialization

```
package main

import (
    "log"
    farewell "temporal101/exercises/farewell-workflow/solution"

    "go.temporal.io/sdk/client"
    "go.temporal.io/sdk/worker"
)

func main() {
    c, err := client.Dial(client.Options{})
    if err != nil {
        log.Fatalln("Unable to create client", err)
    }
    defer c.Close()

    w := worker.New(c, "greeting-tasks", worker.Options{})

    w.RegisterWorkflow(farewell.GreetSomeone)
    w.RegisterActivity(farewell.GreetInSpanish)
    w.RegisterActivity(farewell.FarewellInSpanish)

    err = w.Run(worker.InterruptCh())
    if err != nil {
        log.Fatalln("Unable to start worker", err)
    }
}
```

```
// import statements and unused code omitted from this example
func FarewellInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-farewell", name)
    return greeting, err
}

// utility function for making calls to the microservices
func callService(stem string, name string) (string, error) {
    base := "http://localhost:9999/" + stem + "?name=%s"
    url := fmt.Sprintf(base, url.QueryEscape(name))

    resp, err := http.Get(url)
    if err != nil {
        return "", err
    }
    defer resp.Body.Close()

    body, err := ioutil.ReadAll(resp.Body)
    if err != nil {
        return "", err
    }

    translation := string(body)
    status := resp.StatusCode
    if status >= 400 {
        message := fmt.Sprintf("HTTP Error %d: %s", status, translation)
        return "", errors.New(message)
    }

    return translation, nil
}
```

Event History

WorkflowExecutionStarted	
WorkflowTaskScheduled	
WorkflowTaskStarted	
WorkflowTaskCompleted	
ActivityTaskScheduled	(Greeting)
ActivityTaskStarted	(Greeting)
ActivityTaskCompleted	(Greeting)
WorkflowTaskScheduled	
WorkflowTaskStarted	
WorkflowTaskCompleted	
ActivityTaskScheduled	(Farewell)
ActivityTaskStarted	(Farewell)



Activity Definitions

```go
package farewell    // import statements omitted for brevity

func GreetInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-greeting", name)
    return greeting, err
}

func FarewellInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-farewell", name)
    return greeting, err
}

// utility function for making calls to the microservices
func callService(stem string, name string) (string, error) {
    base := "http://localhost:9999/" + stem + "?name=%s"
    url := fmt.Sprintf(base, url.QueryEscape(name))

    resp, err := http.Get(url)
    if err != nil {
        return "", err
    }
    defer resp.Body.Close()

    body, err := ioutil.ReadAll(resp.Body)
    if err != nil {
        return "", err
    }

    translation := string(body)
    status := resp.StatusCode
    if status >= 400 {
        message := fmt.Sprintf("HTTP Error %d: %s", status, translation)
        return "", errors.New(message)
    }

    return translation, nil
}
```

Workflow Definition

```go
package farewell

import (
    "time"

    "go.temporal.io/sdk/workflow"
)

func GreetSomeone(ctx workflow.Context, name string) (string, error) {
    options := workflow.ActivityOptions{
        StartToCloseTimeout: time.Second * 5,
    }
    ctx = workflow.WithActivityOptions(ctx, options)

    var spanishGreeting string
    err := workflow.ExecuteActivity(ctx, GreetInSpanish, name).Get(ctx, &spanishGreeting)
    if err != nil {
        return "", err
    }

    var spanishFarewell string
    err = workflow.ExecuteActivity(ctx, FarewellInSpanish, name).Get(ctx, &spanishFarewell)
    if err != nil {
        return "", err
    }

    var helloGoodbye = "\n" + spanishGreeting + "\n" + spanishFarewell

    return helloGoodbye, nil
}
```

Worker Initialization

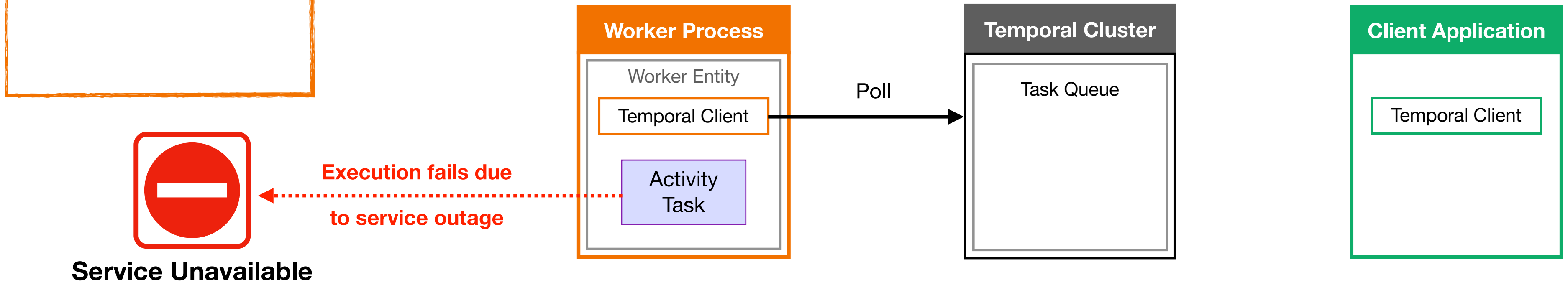
```go
package main

import (
    "log"
    farewell "temporal101/exercises/farewell-workflow/solution"

    "go.temporal.io/sdk/client"
    "go.temporal.io/sdk/worker"
)

func main() {
    c, err := client.Dial(client.Options{})
    if err != nil {
        log.Fatalln("Unable to create client", err)
    }
    defer c.Close()

    w := worker.New(c, "greeting-tasks", worker.Options{})

    w.RegisterWorkflow(farewell.GreetSomeone)
    w.RegisterActivity(farewell.GreetInSpanish)
    w.RegisterActivity(farewell.FarewellInSpanish)

    err = w.Run(worker.InterruptCh())
    if err != nil {
        log.Fatalln("Unable to start worker", err)
    }
}
```

```go
// import statements and unused code omitted from this example
func FarewellInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-farewell", name)
    if err != nil {
        return "", err
    }
    return greeting, nil
}

// utility function for making calls to the microservices
func callService(stem string, name string) (string, error) {
    base := "http://localhost:9999/" + stem + "?name=%s"
    url := fmt.Sprintf(base, url.QueryEscape(name))

    resp, err := http.Get(url)
    if err != nil {
        return "", err
    }
    defer resp.Body.Close()

    body, err := ioutil.ReadAll(resp.Body)
    if err != nil {
        return "", err
    }

    translation := string(body)
    status := resp.StatusCode
    if status >= 400 {
        message := fmt.Sprintf("HTTP Error %d: %s", status, translation)
        return "", errors.New(message)
    }

    return translation, nil
}
```

Error

Event History

WorkflowExecutionStarted	
WorkflowTaskScheduled	
WorkflowTaskStarted	
WorkflowTaskCompleted	
ActivityTaskScheduled	(Greeting)
ActivityTaskStarted	(Greeting)
ActivityTaskCompleted	(Greeting)
WorkflowTaskScheduled	
WorkflowTaskStarted	
WorkflowTaskCompleted	
ActivityTaskScheduled	(Farewell)
ActivityTaskStarted	(Farewell)



Activity Definitions

```
package farewell    // import statements omitted for brevity

func GreetInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-greeting", name)
    return greeting, err
}

func FarewellInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-farewell", name)
    return greeting, err
}

// utility function for making calls to the microservices
func callService(stem string, name string) (string, error) {
    base := "http://localhost:9999/" + stem + "?name=%s"
    url := fmt.Sprintf(base, url.QueryEscape(name))

    resp, err := http.Get(url)
    if err != nil {
        return "", err
    }
    defer resp.Body.Close()

    body, err := ioutil.ReadAll(resp.Body)
    if err != nil {
        return "", err
    }

    translation := string(body)
    status := resp.StatusCode
    if status >= 400 {
        message := fmt.Sprintf("HTTP Error %d: %s", status, translation)
        return "", errors.New(message)
    }

    return translation, nil
}
```

Workflow Definition

```
package farewell

import (
    "time"

    "go.temporal.io/sdk/workflow"
)

func GreetSomeone(ctx workflow.Context, name string) (string, error) {
    options := workflow.ActivityOptions{
        StartToCloseTimeout: time.Second * 5,
    }
    ctx = workflow.WithActivityOptions(ctx, options)

    var spanishGreeting string
    err := workflow.ExecuteActivity(ctx, GreetInSpanish, name).Get(ctx, &spanishGreeting)
    if err != nil {
        return "", err
    }

    var spanishFarewell string
    err = workflow.ExecuteActivity(ctx, FarewellInSpanish, name).Get(ctx, &spanishFarewell)
    if err != nil {
        return "", err
    }

    var helloGoodbye = "\n" + spanishGreeting + "\n" + spanishFarewell

    return helloGoodbye, nil
}
```

Worker Initialization

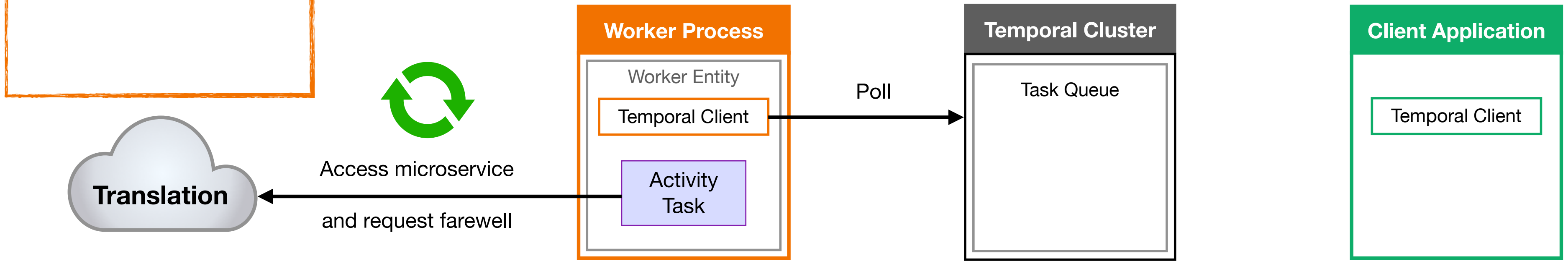
```
package main

import (
    "log"
    farewell "temporal101/exercises/farewell-workflow/solution"

    "go.temporal.io/sdk/client"
    "go.temporal.io/sdk/worker"
)

func main() {
    c, err := client.Dial(client.Options{})
    if err != nil {
        log.Fatalln("Unable to create client", err)
    }
    defer c.Close()

    w := worker.New(c, "greeting-tasks", worker.Options{})

    w.RegisterWorkflow(farewell.GreetSomeone)
    w.RegisterActivity(farewell.GreetInSpanish)
    w.RegisterActivity(farewell.FarewellInSpanish)

    err = w.Run(worker.InterruptCh())
    if err != nil {
        log.Fatalln("Unable to start worker", err)
    }
}
```

```
// import statements and unused code omitted from this example
func FarewellInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-farewell", name)
    if err != nil {
        return "", err
    }
    return greeting, nil
}

// utility function for making calls to the microservices
func callService(stem string, name string) (string, error) {
    base := "http://localhost:9999/" + stem + "?name=%s"
    url := fmt.Sprintf(base, url.QueryEscape(name))

    resp, err := http.Get(url)
    if err != nil {
        return "", err
    }
    defer resp.Body.Close()

    body, err := ioutil.ReadAll(resp.Body)
    if err != nil {
        return "", err
    }

    translation := string(body)
    status := resp.StatusCode
    if status >= 400 {
        message := fmt.Sprintf("HTTP Error %d: %s", status, translation)
        return "", errors.New(message)
    }

    return translation, nil
}
```

Activity is invoked again during retry

Event History

WorkflowExecutionStarted	
WorkflowTaskScheduled	
WorkflowTaskStarted	
WorkflowTaskCompleted	
ActivityTaskScheduled	(Greeting)
ActivityTaskStarted	(Greeting)
ActivityTaskCompleted	(Greeting)
WorkflowTaskScheduled	
WorkflowTaskStarted	
WorkflowTaskCompleted	
ActivityTaskScheduled	(Farewell)
ActivityTaskStarted	(Farewell)



Activity Definitions

```go
package farewell    // import statements omitted for brevity

func GreetInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-greeting", name)
    return greeting, err
}

func FarewellInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-farewell", name)
    return greeting, err
}

// utility function for making calls to the microservices
func callService(stem string, name string) (string, error) {
    base := "http://localhost:9999/" + stem + "?name=%s"
    url := fmt.Sprintf(base, url.QueryEscape(name))

    resp, err := http.Get(url)
    if err != nil {
        return "", err
    }
    defer resp.Body.Close()

    body, err := ioutil.ReadAll(resp.Body)
    if err != nil {
        return "", err
    }

    translation := string(body)
    status := resp.StatusCode
    if status >= 400 {
        message := fmt.Sprintf("HTTP Error %d: %s", status, translation)
        return "", errors.New(message)
    }

    return translation, nil
}
```

Workflow Definition

```go
package farewell

import (
    "time"

    "go.temporal.io/sdk/workflow"
)

func GreetSomeone(ctx workflow.Context, name string) (string, error) {
    options := workflow.ActivityOptions{
        StartToCloseTimeout: time.Second * 5,
    }
    ctx = workflow.WithActivityOptions(ctx, options)

    var spanishGreeting string
    err := workflow.ExecuteActivity(ctx, GreetInSpanish, name).Get(ctx, &spanishGreeting)
    if err != nil {
        return "", err
    }

    var spanishFarewell string
    err = workflow.ExecuteActivity(ctx, FarewellInSpanish, name).Get(ctx, &spanishFarewell)
    if err != nil {
        return "", err
    }

    var helloGoodbye = "\n" + spanishGreeting + "\n" + spanishFarewell

    return helloGoodbye, nil
}
```

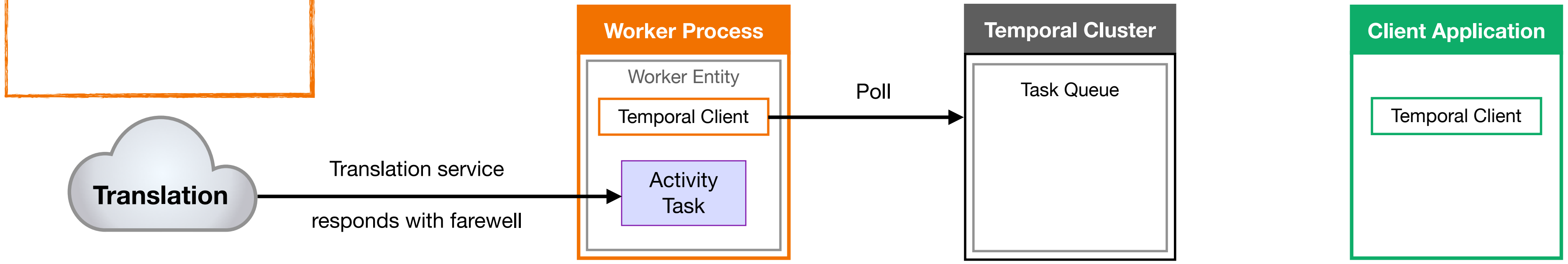
Worker Initialization

```go
package main

import (
    "log"
    farewell "temporal101/exercises/farewell-workflow/solution"

    "go.temporal.io/sdk/client"
    "go.temporal.io/sdk/worker"
)

func main() {
    c, err := client.Dial(client.Options{})
    if err != nil {
        log.Fatalln("Unable to create client", err)
    }
    defer c.Close()

    w := worker.New(c, "greeting-tasks", worker.Options{})

    w.RegisterWorkflow(farewell.GreetSomeone)
    w.RegisterActivity(farewell.GreetInSpanish)
    w.RegisterActivity(farewell.FarewellInSpanish)

    err = w.Run(worker.InterruptCh())
    if err != nil {
        log.Fatalln("Unable to start worker", err)
    }
}
```

```go
// import statements and unused code omitted from this example
func FarewellInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-farewell", name)
    return greeting, err
}

// utility function for making calls to the microservices
func callService(stem string, name string) (string, error) {
    base := "http://localhost:9999/" + stem + "?name=%s"
    url := fmt.Sprintf(base, url.QueryEscape(name))

    resp, err := http.Get(url)
    if err != nil {
        return "", err
    }
    defer resp.Body.Close()

    body, err := ioutil.ReadAll(resp.Body)
    if err != nil {
        return "", err
    }

    translation := string(body)
    status := resp.StatusCode
    if status >= 400 {
        message := fmt.Sprintf("HTTP Error %d: %s", status, translation)
        return "", errors.New(message)
    }

    return translation, nil
}
```

Event History

WorkflowExecutionStarted	
WorkflowTaskScheduled	
WorkflowTaskStarted	
WorkflowTaskCompleted	
ActivityTaskScheduled	(Greeting)
ActivityTaskStarted	(Greeting)
ActivityTaskCompleted	(Greeting)
WorkflowTaskScheduled	
WorkflowTaskStarted	
WorkflowTaskCompleted	
ActivityTaskScheduled	(Farewell)
ActivityTaskStarted	(Farewell)



Activity Definitions

```
package farewell    // import statements omitted for brevity

func GreetInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-greeting", name)
    return greeting, err
}

func FarewellInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-farewell", name)
    return greeting, err
}

// utility function for making calls to the microservices
func callService(stem string, name string) (string, error) {
    base := "http://localhost:9999/" + stem + "?name=%s"
    url := fmt.Sprintf(base, url.QueryEscape(name))

    resp, err := http.Get(url)
    if err != nil {
        return "", err
    }
    defer resp.Body.Close()

    body, err := ioutil.ReadAll(resp.Body)
    if err != nil {
        return "", err
    }

    translation := string(body)
    status := resp.StatusCode
    if status >= 400 {
        message := fmt.Sprintf("HTTP Error %d: %s", status, translation)
        return "", errors.New(message)
    }

    return translation, nil
}
```

Workflow Definition

```
package farewell

import (
    "time"

    "go.temporal.io/sdk/workflow"
)

func GreetSomeone(ctx workflow.Context, name string) (string, error) {
    options := workflow.ActivityOptions{
        StartToCloseTimeout: time.Second * 5,
    }
    ctx = workflow.WithActivityOptions(ctx, options)

    var spanishGreeting string
    err := workflow.ExecuteActivity(ctx, GreetInSpanish, name).Get(ctx, &spanishGreeting)
    if err != nil {
        return "", err
    }

    var spanishFarewell string
    err = workflow.ExecuteActivity(ctx, FarewellInSpanish, name).Get(ctx, &spanishFarewell)
    if err != nil {
        return "", err
    }

    var helloGoodbye = "\n" + spanishGreeting + "\n" + spanishFarewell

    return helloGoodbye, nil
}
```

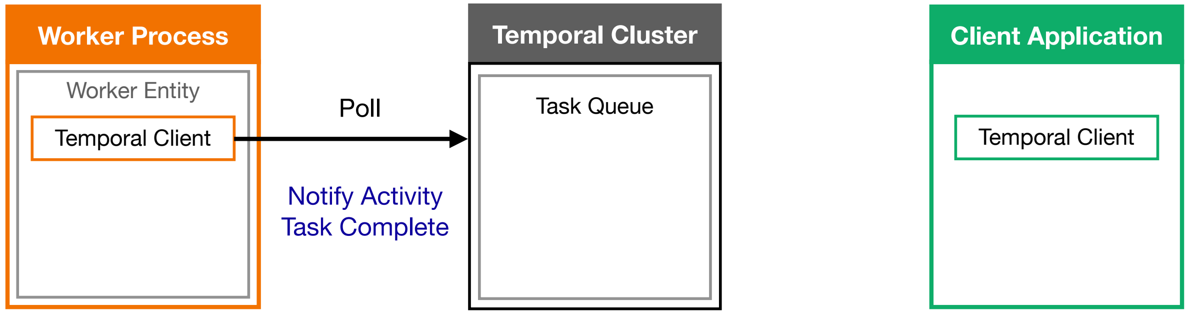
Worker Initialization

```
package main

import (
    "log"
    farewell "temporal101/exercises/farewell-workflow/solution"

    "go.temporal.io/sdk/client"
    "go.temporal.io/sdk/worker"
)

func main() {
    c, err := client.Dial(client.Options{})
    if err != nil {
        log.Fatalln("Unable to create client", err)
    }
    defer c.Close()

    w := worker.New(c, "greeting-tasks", worker.Options{})

    w.RegisterWorkflow(farewell.GreetSomeone)
    w.RegisterActivity(farewell.GreetInSpanish)
    w.RegisterActivity(farewell.FarewellInSpanish)

    err = w.Run(worker.InterruptCh())
    if err != nil {
        log.Fatalln("Unable to start worker", err)
    }
}
```

```
// import statements and unused code omitted from this example
func FarewellInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-farewell", name)
    if err != nil {
        return "", err
    }
    return greeting, nil
}

// utility function for making calls to the microservices
func callService(stem string, name string) (string, error) {
    base := "http://localhost:9999/" + stem + "?name=%s"
    url := fmt.Sprintf(base, url.QueryEscape(name))

    resp, err := http.Get(url)
    if err != nil {
        return "", err
    }
    defer resp.Body.Close()

    body, err := ioutil.ReadAll(resp.Body)
    if err != nil {
        return "", err
    }

    translation := string(body)
    status := resp.StatusCode
    if status >= 400 {
        message := fmt.Sprintf("HTTP Error %d: %s", status, translation)
        return "", errors.New(message)
    }

    return translation, nil
}
```

Event History

WorkflowExecutionStarted	
WorkflowTaskScheduled	
WorkflowTaskStarted	
WorkflowTaskCompleted	
ActivityTaskScheduled	(Greeting)
ActivityTaskStarted	(Greeting)
ActivityTaskCompleted	(Greeting)
WorkflowTaskScheduled	
WorkflowTaskStarted	
WorkflowTaskCompleted	
ActivityTaskScheduled	(Farewell)
ActivityTaskStarted	(Farewell)
ActivityTaskCompleted	(Farewell)



Activity Definitions

```go
package farewell    // import statements omitted for brevity

func GreetInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-greeting", name)
    return greeting, err
}

func FarewellInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-farewell", name)
    return greeting, err
}

// utility function for making calls to the microservices
func callService(stem string, name string) (string, error) {
    base := "http://localhost:9999/" + stem + "?name=%s"
    url := fmt.Sprintf(base, url.QueryEscape(name))

    resp, err := http.Get(url)
    if err != nil {
        return "", err
    }
    defer resp.Body.Close()

    body, err := ioutil.ReadAll(resp.Body)
    if err != nil {
        return "", err
    }

    translation := string(body)
    status := resp.StatusCode
    if status >= 400 {
        message := fmt.Sprintf("HTTP Error %d: %s", status, translation)
        return "", errors.New(message)
    }

    return translation, nil
}
```

Workflow Definition

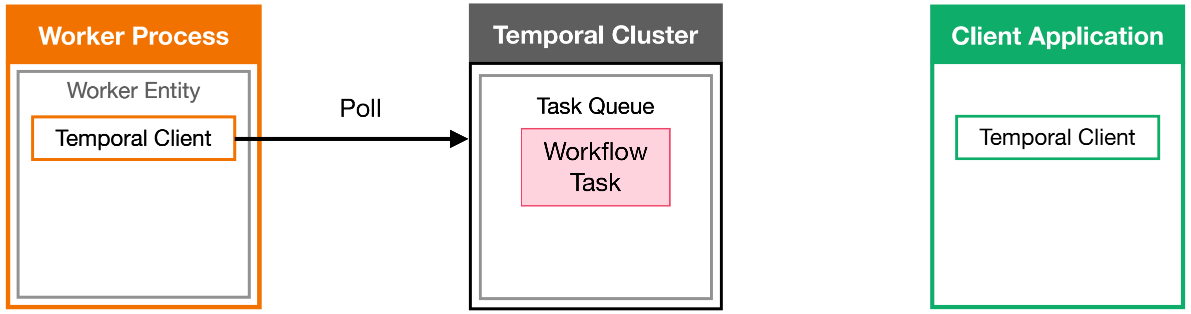
```go
package farewell

import (
    "time"

    "go.temporal.io/sdk/workflow"
)

func GreetSomeone(ctx workflow.Context, name string) (string, error) {
    options := workflow.ActivityOptions{
        StartToCloseTimeout: time.Second * 5,
    }
    ctx = workflow.WithActivityOptions(ctx, options)

    var spanishGreeting string
    err := workflow.ExecuteActivity(ctx, GreetInSpanish, name).Get(ctx, &spanishGreeting)
    if err != nil {
        return "", err
    }

    var spanishFarewell string
    err = workflow.ExecuteActivity(ctx, FarewellInSpanish, name).Get(ctx, &spanishFarewell)
    if err != nil {
        return "", err
    }

    var helloGoodbye = "\n" + spanishGreeting + "\n" + spanishFarewell

    return helloGoodbye, nil
}
```

Worker Initialization

```go
package main

import (
    "log"
    farewell "temporal101/exercises/farewell-workflow/solution"

    "go.temporal.io/sdk/client"
    "go.temporal.io/sdk/worker"
)

func main() {
    c, err := client.Dial(client.Options{})
    if err != nil {
        log.Fatalln("Unable to create client", err)
    }
    defer c.Close()

    w := worker.New(c, "greeting-tasks", worker.Options{})

    w.RegisterWorkflow(farewell.GreetSomeone)
    w.RegisterActivity(farewell.GreetInSpanish)
    w.RegisterActivity(farewell.FarewellInSpanish)

    err = w.Run(worker.InterruptCh())
    if err != nil {
        log.Fatalln("Unable to start worker", err)
    }
}
```

Event History

WorkflowExecutionStarted	
WorkflowTaskScheduled	
WorkflowTaskStarted	
WorkflowTaskCompleted	
ActivityTaskScheduled	(Greeting)
ActivityTaskStarted	(Greeting)
ActivityTaskCompleted	(Greeting)
WorkflowTaskScheduled	
WorkflowTaskStarted	
WorkflowTaskCompleted	
ActivityTaskScheduled	(Farewell)
ActivityTaskStarted	(Farewell)
ActivityTaskCompleted	(Farewell)
WorkflowTaskScheduled	



Activity Definitions

```go
package farewell    // import statements omitted for brevity

func GreetInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-greeting", name)
    return greeting, err
}

func FarewellInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-farewell", name)
    return greeting, err
}

// utility function for making calls to the microservices
func callService(stem string, name string) (string, error) {
    base := "http://localhost:9999/" + stem + "?name=%s"
    url := fmt.Sprintf(base, url.QueryEscape(name))

    resp, err := http.Get(url)
    if err != nil {
        return "", err
    }
    defer resp.Body.Close()

    body, err := ioutil.ReadAll(resp.Body)
    if err != nil {
        return "", err
    }

    translation := string(body)
    status := resp.StatusCode
    if status >= 400 {
        message := fmt.Sprintf("HTTP Error %d: %s", status, translation)
        return "", errors.New(message)
    }

    return translation, nil
}
```

Workflow Definition

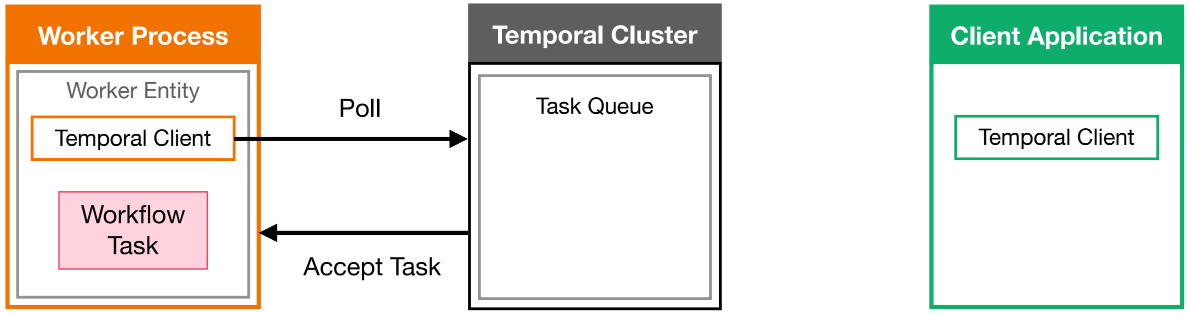
```go
package farewell

import (
    "time"

    "go.temporal.io/sdk/workflow"
)

func GreetSomeone(ctx workflow.Context, name string) (string, error) {
    options := workflow.ActivityOptions{
        StartToCloseTimeout: time.Second * 5,
    }
    ctx = workflow.WithActivityOptions(ctx, options)

    var spanishGreeting string
    err := workflow.ExecuteActivity(ctx, GreetInSpanish, name).Get(ctx, &spanishGreeting)
    if err != nil {
        return "", err
    }

    var spanishFarewell string
    err = workflow.ExecuteActivity(ctx, FarewellInSpanish, name).Get(ctx, &spanishFarewell)
    if err != nil {
        return "", err
    }

    var helloGoodbye = "\n" + spanishGreeting + "\n" + spanishFarewell

    return helloGoodbye, nil
}
```

Worker Initialization

```go
package main

import (
    "log"
    farewell "temporal101/exercises/farewell-workflow/solution"

    "go.temporal.io/sdk/client"
    "go.temporal.io/sdk/worker"
)

func main() {
    c, err := client.Dial(client.Options{})
    if err != nil {
        log.Fatalln("Unable to create client", err)
    }
    defer c.Close()

    w := worker.New(c, "greeting-tasks", worker.Options{})

    w.RegisterWorkflow(farewell.GreetSomeone)
    w.RegisterActivity(farewell.GreetInSpanish)
    w.RegisterActivity(farewell.FarewellInSpanish)

    err = w.Run(worker.InterruptCh())
    if err != nil {
        log.Fatalln("Unable to start worker", err)
    }
}
```

Event History

`WorkflowExecutionStarted`	
`WorkflowTaskScheduled`	
`WorkflowTaskStarted`	
`WorkflowTaskCompleted`	
`ActivityTaskScheduled`	(Greeting)
`ActivityTaskStarted`	(Greeting)
`ActivityTaskCompleted`	(Greeting)
`WorkflowTaskScheduled`	
`WorkflowTaskStarted`	
`WorkflowTaskCompleted`	
`ActivityTaskScheduled`	(Farewell)
`ActivityTaskStarted`	(Farewell)
`ActivityTaskCompleted`	(Farewell)
`WorkflowTaskScheduled`	
`WorkflowTaskStarted`	



Activity Definitions

```go
package farewell    // import statements omitted for brevity

func GreetInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-greeting", name)
    return greeting, err
}

func FarewellInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-farewell", name)
    return greeting, err
}

// utility function for making calls to the microservices
func callService(stem string, name string) (string, error) {
    base := "http://localhost:9999/" + stem + "?name=%s"
    url := fmt.Sprintf(base, url.QueryEscape(name))

    resp, err := http.Get(url)
    if err != nil {
        return "", err
    }
    defer resp.Body.Close()

    body, err := ioutil.ReadAll(resp.Body)
    if err != nil {
        return "", err
    }

    translation := string(body)
    status := resp.StatusCode
    if status >= 400 {
        message := fmt.Sprintf("HTTP Error %d: %s", status, translation)
        return "", errors.New(message)
    }

    return translation, nil
}
```

Workflow Definition

```go
package farewell

import (
    "time"

    "go.temporal.io/sdk/workflow"
)

func GreetSomeone(ctx workflow.Context, name string) (string, error) {
    options := workflow.ActivityOptions{
        StartToCloseTimeout: time.Second * 5,
    }
    ctx = workflow.WithActivityOptions(ctx, options)

    var spanishGreeting string
    err := workflow.ExecuteActivity(ctx, GreetInSpanish, name).Get(ctx, &spanishGreeting)
    if err != nil {
        return "", err
    }

    var spanishFarewell string
    err = workflow.ExecuteActivity(ctx, FarewellInSpanish, name).Get(ctx, &spanishFarewell)
    if err != nil {
        return "", err
    }

    var helloGoodbye = "\n" + spanishGreeting + "\n" + spanishFarewell

    return helloGoodbye, nil
}
```

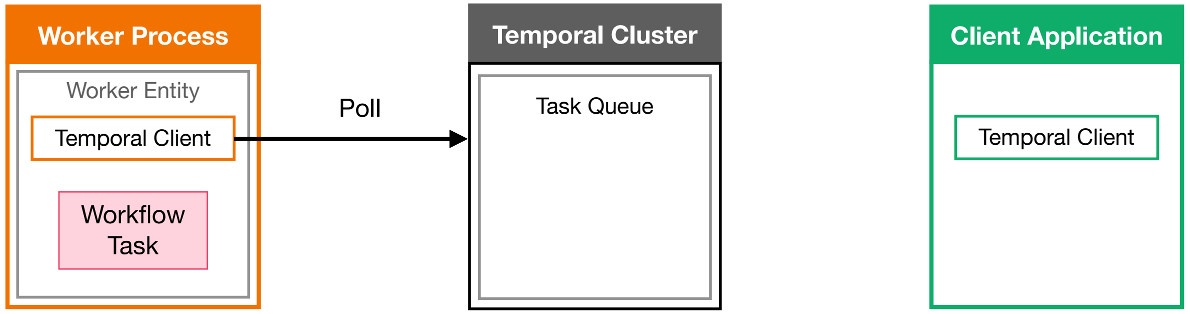
Worker Initialization

```go
package main

import (
    "log"
    farewell "temporal101/exercises/farewell-workflow/solution"

    "go.temporal.io/sdk/client"
    "go.temporal.io/sdk/worker"
)

func main() {
    c, err := client.Dial(client.Options{})
    if err != nil {
        log.Fatalln("Unable to create client", err)
    }
    defer c.Close()

    w := worker.New(c, "greeting-tasks", worker.Options{})

    w.RegisterWorkflow(farewell.GreetSomeone)
    w.RegisterActivity(farewell.GreetInSpanish)
    w.RegisterActivity(farewell.FarewellInSpanish)

    err = w.Run(worker.InterruptCh())
    if err != nil {
        log.Fatalln("Unable to start worker", err)
    }
}
```

```go
// ... code above has been omitted from this excerpt


func GreetSomeone(ctx workflow.Context, name string) (string, error) {
    options := workflow.ActivityOptions{
        StartToCloseTimeout: time.Second * 5,
    }
    ctx = workflow.WithActivityOptions(ctx, options)

    var spanishGreeting string
    err := workflow.ExecuteActivity(ctx, GreetInSpanish, name).Get(ctx, &spanishGreeting)
    if err != nil {
        return "", err
    }

    var spanishFarewell string
    err = workflow.ExecuteActivity(ctx, FarewellInSpanish, name).Get(ctx, &spanishFarewell)
    if err != nil {
        return "", err
    }

    var helloGoodbye = "\n" + spanishGreeting + "\n" + spanishFarewell

    return helloGoodbye, nil
}
```

Event History

WorkflowExecutionStarted	
WorkflowTaskScheduled	
WorkflowTaskStarted	
WorkflowTaskCompleted	
ActivityTaskScheduled	(Greeting)
ActivityTaskStarted	(Greeting)
ActivityTaskCompleted	(Greeting)
WorkflowTaskScheduled	
WorkflowTaskStarted	
WorkflowTaskCompleted	
ActivityTaskScheduled	(Farewell)
ActivityTaskStarted	(Farewell)
ActivityTaskCompleted	(Farewell)
WorkflowTaskScheduled	
WorkflowTaskStarted	



Activity Definitions

```
package farewell    // import statements omitted for brevity

func GreetInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-greeting", name)
    return greeting, err
}

func FarewellInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-farewell", name)
    return greeting, err
}

// utility function for making calls to the microservices
func callService(stem string, name string) (string, error) {
    base := "http://localhost:9999/" + stem + "?name=%s"
    url := fmt.Sprintf(base, url.QueryEscape(name))

    resp, err := http.Get(url)
    if err != nil {
        return "", err
    }
    defer resp.Body.Close()

    body, err := ioutil.ReadAll(resp.Body)
    if err != nil {
        return "", err
    }

    translation := string(body)
    status := resp.StatusCode
    if status >= 400 {
        message := fmt.Sprintf("HTTP Error %d: %s", status, translation)
        return "", errors.New(message)
    }

    return translation, nil
}
```

Workflow Definition

```
package farewell

import (
    "time"

    "go.temporal.io/sdk/workflow"
)

func GreetSomeone(ctx workflow.Context, name string) (string, error) {
    options := workflow.ActivityOptions{
        StartToCloseTimeout: time.Second * 5,
    }
    ctx = workflow.WithActivityOptions(ctx, options)

    var spanishGreeting string
    err := workflow.ExecuteActivity(ctx, GreetInSpanish, name).Get(ctx, &spanishGreeting)
    if err != nil {
        return "", err
    }

    var spanishFarewell string
    err = workflow.ExecuteActivity(ctx, FarewellInSpanish, name).Get(ctx, &spanishFarewell)
    if err != nil {
        return "", err
    }

    var helloGoodbye = "\n" + spanishGreeting + "\n" + spanishFarewell

    return helloGoodbye, nil
}
```

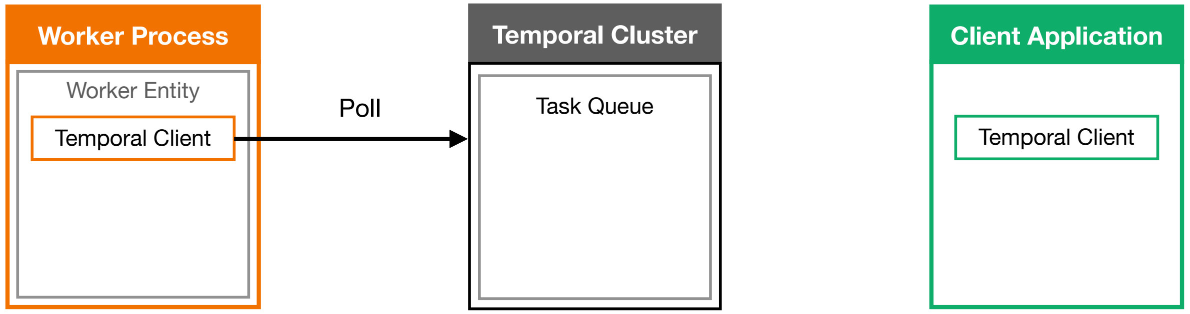
Worker Initialization

```
package main

import (
    "log"
    farewell "temporal101/exercises/farewell-workflow/solution"

    "go.temporal.io/sdk/client"
    "go.temporal.io/sdk/worker"
)

func main() {
    c, err := client.Dial(client.Options{})
    if err != nil {
        log.Fatalln("Unable to create client", err)
    }
    defer c.Close()

    w := worker.New(c, "greeting-tasks", worker.Options{})

    w.RegisterWorkflow(farewell.GreetSomeone)
    w.RegisterActivity(farewell.GreetInSpanish)
    w.RegisterActivity(farewell.FarewellInSpanish)

    err = w.Run(worker.InterruptCh())
    if err != nil {
        log.Fatalln("Unable to start worker", err)
    }
}
```

```
// ... code above has been omitted from this excerpt


func GreetSomeone(ctx workflow.Context, name string) (string, error) {
    options := workflow.ActivityOptions{
        StartToCloseTimeout: time.Second * 5,
    }
    ctx = workflow.WithActivityOptions(ctx, options)

    var spanishGreeting string
    err := workflow.ExecuteActivity(ctx, GreetInSpanish, name).Get(ctx, &spanishGreeting)
    if err != nil {
        return "", err
    }

    var spanishFarewell string
    err = workflow.ExecuteActivity(ctx, FarewellInSpanish, name).Get(ctx, &spanishFarewell)
    if err != nil {
        return "", err
    }

    var helloGoodbye = "\n" + spanishGreeting + "\n" + spanishFarewell

    return helloGoodbye, nil
}
```

Event History

WorkflowExecutionStarted	
WorkflowTaskScheduled	
WorkflowTaskStarted	
WorkflowTaskCompleted	
ActivityTaskScheduled	(Greeting)
ActivityTaskStarted	(Greeting)
ActivityTaskCompleted	(Greeting)
WorkflowTaskScheduled	
WorkflowTaskStarted	
WorkflowTaskCompleted	
ActivityTaskScheduled	(Farewell)
ActivityTaskStarted	(Farewell)
ActivityTaskCompleted	(Farewell)
WorkflowTaskScheduled	
WorkflowTaskStarted	
WorkflowTaskCompleted	



Activity Definitions

```
package farewell    // import statements omitted for brevity

func GreetInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-greeting", name)
    return greeting, err
}

func FarewellInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-farewell", name)
    return greeting, err
}

// utility function for making calls to the microservices
func callService(stem string, name string) (string, error) {
    base := "http://localhost:9999/" + stem + "?name=%s"
    url := fmt.Sprintf(base, url.QueryEscape(name))

    resp, err := http.Get(url)
    if err != nil {
        return "", err
    }
    defer resp.Body.Close()

    body, err := ioutil.ReadAll(resp.Body)
    if err != nil {
        return "", err
    }

    translation := string(body)
    status := resp.StatusCode
    if status >= 400 {
        message := fmt.Sprintf("HTTP Error %d: %s", status, translation)
        return "", errors.New(message)
    }

    return translation, nil
}
```

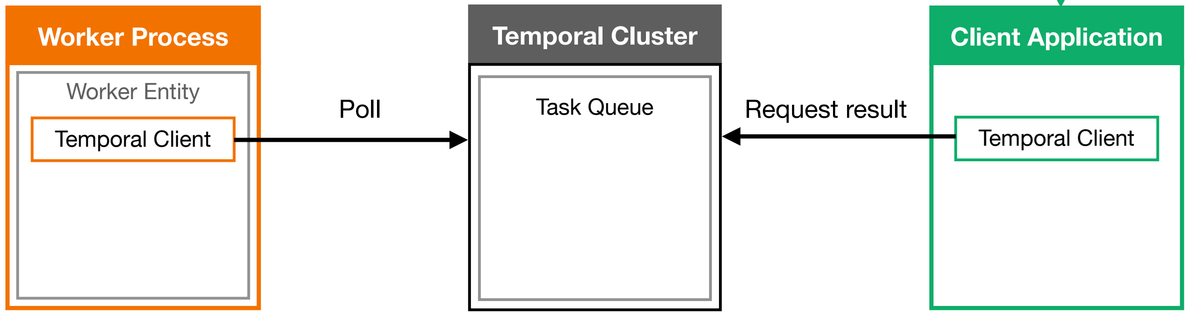
Workflow Definition

```
package farewell

import (
    "time"

    "go.temporal.io/sdk/workflow"
)

func GreetSomeone(ctx workflow.Context, name string) (string, error) {
    options := workflow.ActivityOptions{
        StartToCloseTimeout: time.Second * 5,
    }
    ctx = workflow.WithActivityOptions(ctx, options)

    var spanishGreeting string
    err := workflow.ExecuteActivity(ctx, GreetInSpanish, name).Get(ctx, &spanishGreeting)
    if err != nil {
        return "", err
    }

    var spanishFarewell string
    err = workflow.ExecuteActivity(ctx, FarewellInSpanish, name).Get(ctx, &spanishFarewell)
    if err != nil {
        return "", err
    }

    var helloGoodbye = "\n" + spanishGreeting + "\n" + spanishFarewell

    return helloGoodbye, nil
}
```

Worker Initialization

```
package main

import (
    "log"
    farewell "temporal101/exercises/farewell-workflow/solution"

    "go.temporal.io/sdk/client"
    "go.temporal.io/sdk/worker"
)

func main() {
    c, err := client.Dial(client.Options{})
    if err != nil {
        log.Fatalln("Unable to create client", err)
    }
    defer c.Close()

    w := worker.New(c, "greeting-tasks", worker.Options{})

    w.RegisterWorkflow(farewell.GreetSomeone)
    w.RegisterActivity(farewell.GreetInSpanish)
    w.RegisterActivity(farewell.FarewellInSpanish)

    err = w.Run(worker.InterruptCh())
    if err != nil {
        log.Fatalln("Unable to start worker", err)
    }
}
```

```
// ... this is code within your own application (for example, a web application, mobile app, etc.)


options := client.StartWorkflowOptions{
    ID:        "greeting-workflow",
    TaskQueue: "greeting-tasks",
}

we, err := c.ExecuteWorkflow(context.Background(), options, farewell.GreetSomeone, os.Args[1])
if err != nil {
    log.Fatalln("Unable to execute workflow", err)
}
log.Println("Started workflow", "WorkflowID", we.GetID(), "RunID", we.GetRunID())

var result string
err = we.Get(context.Background(), &result)
if err != nil {
    log.Fatalln("Unable get workflow result", err)
}
log.Println("Workflow result:", result)


// ... other application-specific code might follow
```



Event History

WorkflowExecutionStarted	
WorkflowTaskScheduled	
WorkflowTaskStarted	
WorkflowTaskCompleted	
ActivityTaskScheduled	(Greeting)
ActivityTaskStarted	(Greeting)
ActivityTaskCompleted	(Greeting)
WorkflowTaskScheduled	
WorkflowTaskStarted	
WorkflowTaskCompleted	
ActivityTaskScheduled	(Farewell)
ActivityTaskStarted	(Farewell)
ActivityTaskCompleted	(Farewell)
WorkflowTaskScheduled	
WorkflowTaskStarted	
WorkflowTaskCompleted	
WorkflowExecutionCompleted	

Activity Definitions

```
package farewell    // import statements omitted for brevity

func GreetInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-greeting", name)
    return greeting, err
}

func FarewellInSpanish(ctx context.Context, name string) (string, error) {
    greeting, err := callService("get-spanish-farewell", name)
    return greeting, err
}

// utility function for making calls to the microservices
func callService(stem string, name string) (string, error) {
    base := "http://localhost:9999/" + stem + "?name=%s"
    url := fmt.Sprintf(base, url.QueryEscape(name))

    resp, err := http.Get(url)
    if err != nil {
        return "", err
    }
    defer resp.Body.Close()

    body, err := ioutil.ReadAll(resp.Body)
    if err != nil {
        return "", err
    }

    translation := string(body)
    status := resp.StatusCode
    if status >= 400 {
        message := fmt.Sprintf("HTTP Error %d: %s", status, translation)
        return "", errors.New(message)
    }

    return translation, nil
}
```

Workflow Definition

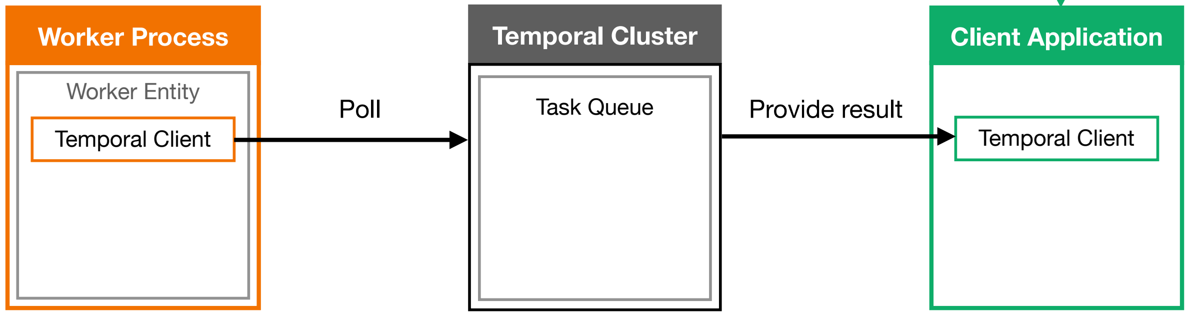
```
package farewell

import (
    "time"

    "go.temporal.io/sdk/workflow"
)

func GreetSomeone(ctx workflow.Context, name string) (string, error) {
    options := workflow.ActivityOptions{
        StartToCloseTimeout: time.Second * 5,
    }
    ctx = workflow.WithActivityOptions(ctx, options)

    var spanishGreeting string
    err := workflow.ExecuteActivity(ctx, GreetInSpanish, name).Get(ctx, &spanishGreeting)
    if err != nil {
        return "", err
    }

    var spanishFarewell string
    err = workflow.ExecuteActivity(ctx, FarewellInSpanish, name).Get(ctx, &spanishFarewell)
    if err != nil {
        return "", err
    }

    var helloGoodbye = "\n" + spanishGreeting + "\n" + spanishFarewell

    return helloGoodbye, nil
}
```

Worker Initialization

```
package main

import (
    "log"
    farewell "temporal101/exercises/farewell-workflow/solution"

    "go.temporal.io/sdk/client"
    "go.temporal.io/sdk/worker"
)

func main() {
    c, err := client.Dial(client.Options{})
    if err != nil {
        log.Fatalln("Unable to create client", err)
    }
    defer c.Close()

    w := worker.New(c, "greeting-tasks", worker.Options{})

    w.RegisterWorkflow(farewell.GreetSomeone)
    w.RegisterActivity(farewell.GreetInSpanish)
    w.RegisterActivity(farewell.FarewellInSpanish)

    err = w.Run(worker.InterruptCh())
    if err != nil {
        log.Fatalln("Unable to start worker", err)
    }
}
```

The End

```
// ... this is code within your own application (for example, a web application, mobile app, etc.)


options := client.StartWorkflowOptions{
    ID:         "greeting-workflow",
    TaskQueue: "greeting-tasks",
}

we, err := c.ExecuteWorkflow(context.Background(), options, farewell.GreetSomeone, os.Args[1])
if err != nil {
    log.Fatalln("Unable to execute workflow", err)
}
log.Println("Started workflow", "WorkflowID", we.GetID(), "RunID", we.GetRunID())

var result string
err = we.Get(context.Background(), &result)
if err != nil {
    log.Fatalln("Unable get workflow result", err)
}
log.Println("Workflow result:", result)

// ... other application-specific code might follow
```



Event History

WorkflowExecutionStarted	
WorkflowTaskScheduled	
WorkflowTaskStarted	
WorkflowTaskCompleted	
ActivityTaskScheduled	(Greeting)
ActivityTaskStarted	(Greeting)
ActivityTaskCompleted	(Greeting)
WorkflowTaskScheduled	
WorkflowTaskStarted	
WorkflowTaskCompleted	
ActivityTaskScheduled	(Farewell)
ActivityTaskStarted	(Farewell)
ActivityTaskCompleted	(Farewell)
WorkflowTaskScheduled	
WorkflowTaskStarted	
WorkflowTaskCompleted	
WorkflowExecutionCompleted	

Temporal 101

00 About this Workshop

01 What is Temporal?

02 Developing a Workflow

03 Executing a Workflow

04 Viewing Workflow Execution History

05 Modifying an Existing Workflow

06 Developing an Activity

07 Handling Activity Failure

08 Understanding Workflow Execution

09 Conclusion

Conclusion (1)

- **Temporal guarantees the durable execution of your applications**
  - In Temporal, Workflows are defined through code (using a Temporal SDK)
- **Temporal Clusters orchestrate code execution**
  - Workers are responsible for actually executing the code
- **The Temporal Cluster maintains dynamically-created task queues**
  - Workers continuously poll a task queue and accept tasks if they have spare capacity
  - You can increase application scalability by adding more Workers
  - You must restart Workers after deploying a code change

Conclusion (2)

- **There are multiple ways of deploying a self-hosted Temporal cluster**
  - Temporal Cloud is an alternative to hosting your own cluster
  - Migrating to / from Temporal Cloud requires little change to application code
- **Namespaces are used for isolation within a cluster**
  - The name is often chosen to indicate a specific team, department, or other category
- **In the Go SDK, a Temporal Workflow is defined through a function**
  - Activities are also defined through functions

Conclusion (3)

- **Activities encapsulate unreliable or non-deterministic code**
  - They are automatically retried upon failure
  - You can change this behavior with a custom Retry Policy
- **The Web UI is a powerful tool for gaining insight into your application**
  - It displays current and recent Workflow Executions
  - The Web UI shows inputs, outputs, and event history

For More Information

- Temporal Documentation
- Temporal Community Forums
- Temporal Community Slack
- Temporal Samples Repositories at GitHub
- Temporal Education Site
- Temporal YouTube channel
- Temporal Community Events

Exercise #4: Finale Workflow

- **During this exercise, you will**
  - Observe that a Workflow and its Activities can be implemented in different languages
    - This example provides a Java Activity and a Go Workflow for you to run
- **Refer to the README.md file in the exercise environment for details**
  - The code is below the **`exercises/finale-workflow`** directory

Thank You