

7 Temporal 101



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 |

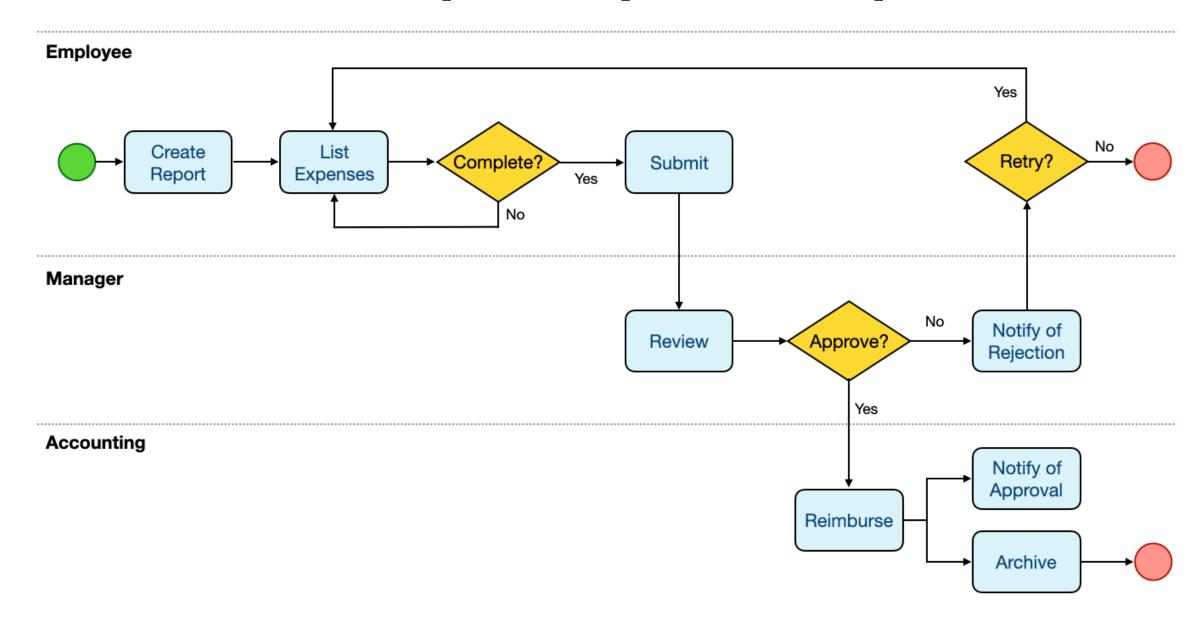
Introducing Temporal

- 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

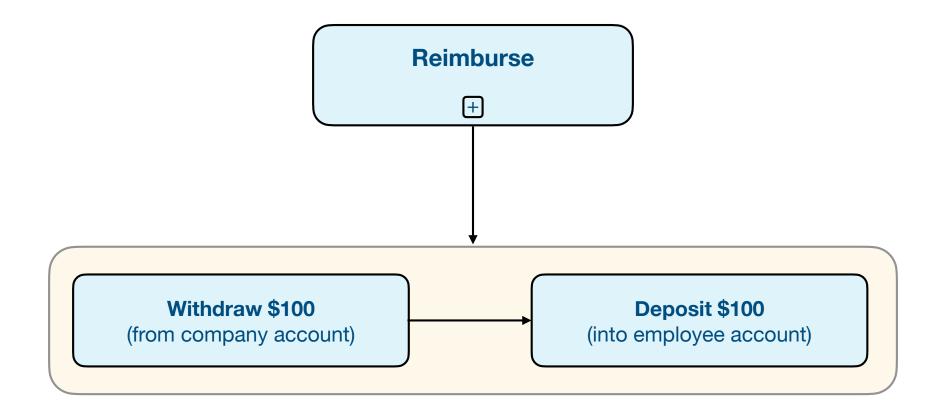
What is a Workflow?

- 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

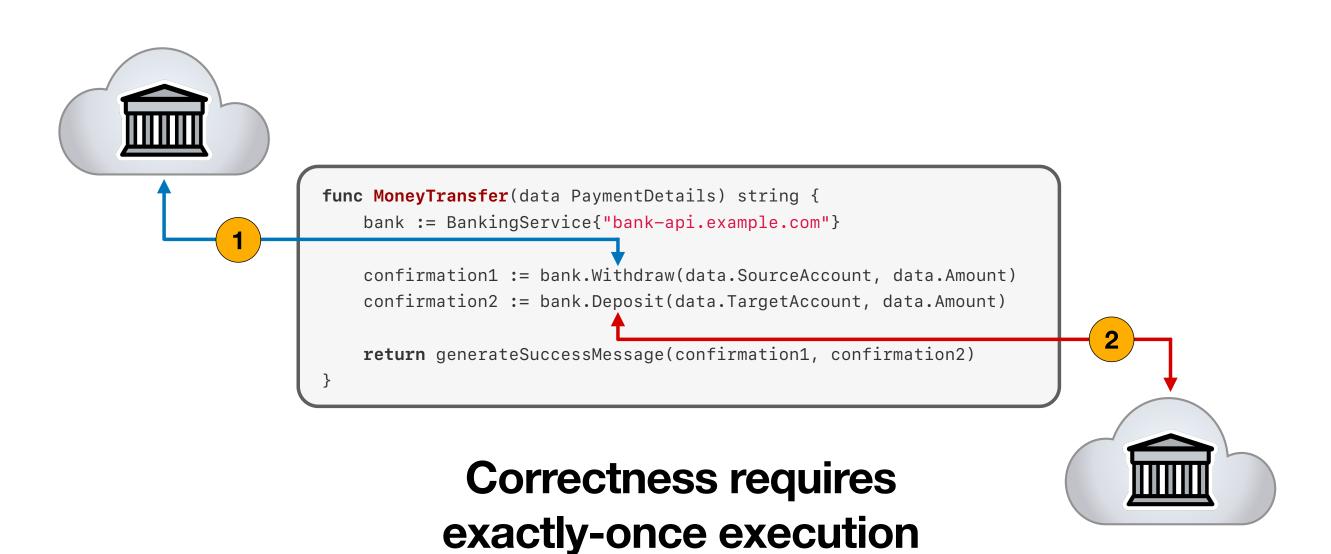


Workflow Example: Reimbursement



Correctness requires exactly-once execution

This Workflow Is a Distributed System



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++ {</pre>
        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++ {</pre>
        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

```
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++ {</pre>
       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++ {</pre>
       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++ {</pre>
           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

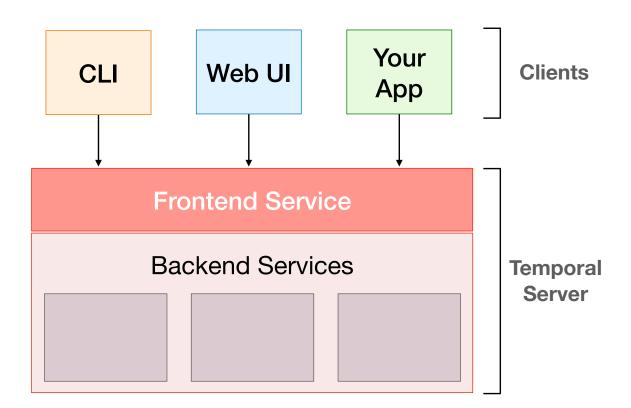
After adding support for request timeouts

```
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++ {</pre>
       confirmation1 = doWithdraw(bank, data.SourceAccount, data.Amount, TIMEOUT_SECONDS)
       if confirmation1 != "FAIL" && confirmation1 != "TIMEOUT" {
           hreak
   if confirmation1 == "" || confirmation1 == "FAIL" {
       return "FAIL: could not withdraw money from source account"
   var confirmation2 = ""
   for attempt := 0; attempt <= MAX_RETRY_ATTEMPTS; attempt++ {</pre>
        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++ {</pre>
           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
   \textbf{return} \texttt{ generateSuccessMessage(confirmation1, confirmation2)}
func doWithdraw(bank BankingService, account string, amount int, timeout time.Duration) string {
   wdReqChannel := make(chan string, 1)
       wdReqChannel <- bank.Withdraw(account, amount)</pre>
   select {
   case confirmation := <-wdReqChannel:</pre>
      return confirmation
   case <-time.After(timeout):</pre>
       return "TIMEOUT
func doDeposit(bank BankingService, account string, amount int, timeout time.Duration) string {
   depReqChannel := make(chan string, 1)
       depReqChannel <- bank.Deposit(account, amount)</pre>
   case confirmation := <-depReqChannel:</pre>
       return confirmation
   case <-time.After(timeout):</pre>
       return "TIMEOUT"
```

Architectural Overview: Temporal Server

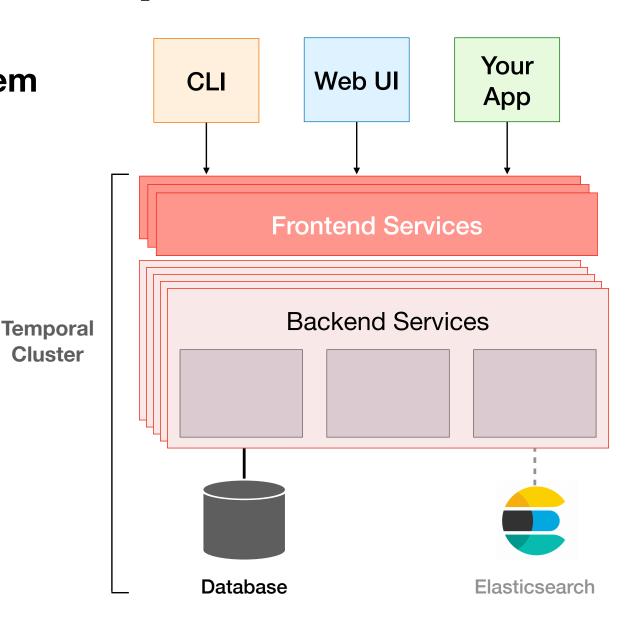
Consists of multiple services

- 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

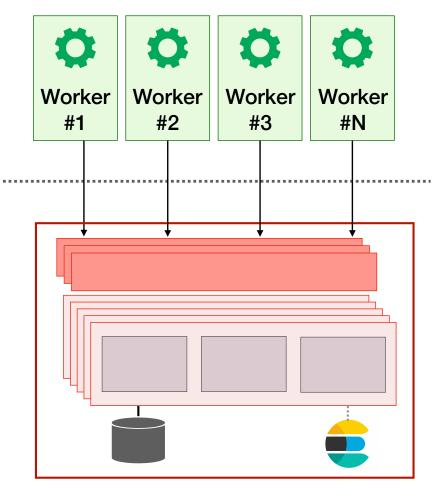
- Temporal Cluster is a complete system
 - It is a deployment of the Temporal Server software and the components used with it
 - A database is a required component
 - Persists Workflow state and Event History
 - Also stores data for durable timers and queues
 - Elasticsearch is an optional component
 - 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

Application Servers



Temporal Cluster

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 (tct1)

- 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

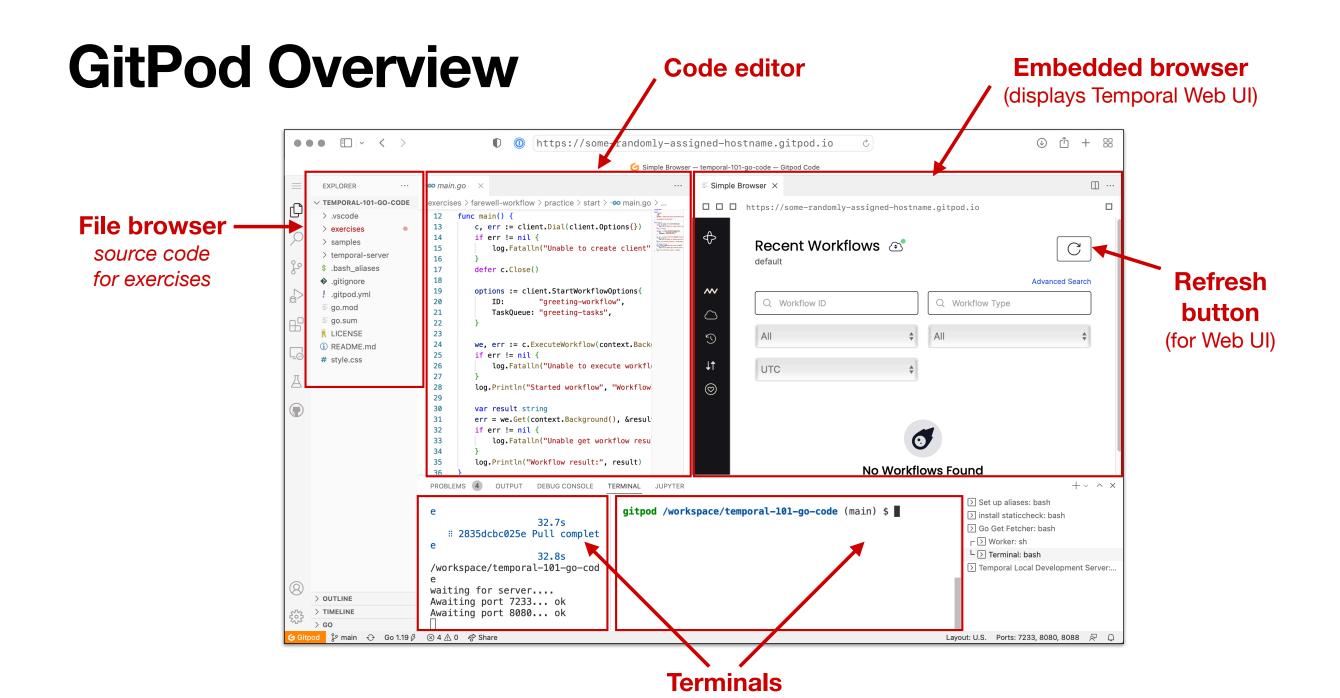
```
$ tctl --help
NAME:
   tctl - A command-line tool for Temporal users
USAGE:
   tctl [global options] command [command options...
VERSION:
   1.18.0
COMMANDS:
                      Operate Temporal namespace
   namespace, n
                      Operate Temporal workflow
   workflow, wf
   activity, 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





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

```
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

```
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)

```
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
 - 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

```
import (
   "app"
   "loa"
   "go.temporal.io/sdk/client"
   "go.temporal.io/sdk/worker"
func main() {
   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()) <---</pre>
   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 **tct1** 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"
    "loa"
    "app"
    "os"
    "go.temporal.io/sdk/client"
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{
              "my-first-workflow",
   TaskQueue: "greeting-tasks",
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 |
| 1 | Wiednynig an Externig Werkiew |
| 06 | Developing an Activity |
| 06 07 | |
| | Developing an Activity |

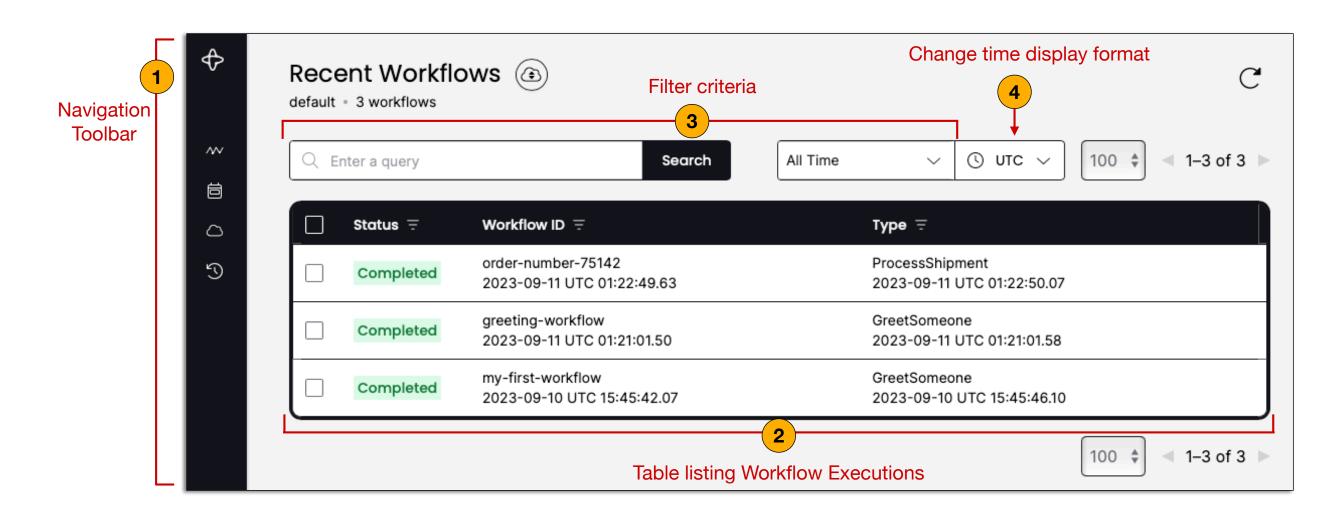
Viewing Workflow History with tctl

```
$ 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}
    WorkflowTaskScheduled
                                 {TaskQueue:{Name:greeting-tasks,
                                  Kind:Normal},
                                  StartToCloseTimeout:10s,
                                  Attempt:1}
                                 {ScheduledEventId:2, Identity:93592@twwmbp@,
    WorkflowTaskStarted
                                  RequestId:10535889-9c10-4073-b38f-4876bbae4db3,
                                  SuggestContinueAsNew:false, HistorySizeBytes:0}
    WorkflowTaskCompleted
                                 {ScheduledEventId:2, StartedEventId:3,
                                  Identity:93592@twwmbp@,
                                  BinaryChecksum: 202d5177234b6ec7b33e3de1b92f2f5f}
    WorkflowExecutionCompleted
                                 {Result:["Hello Donna!"],
                                  WorkflowTaskCompletedEventId:4}
```

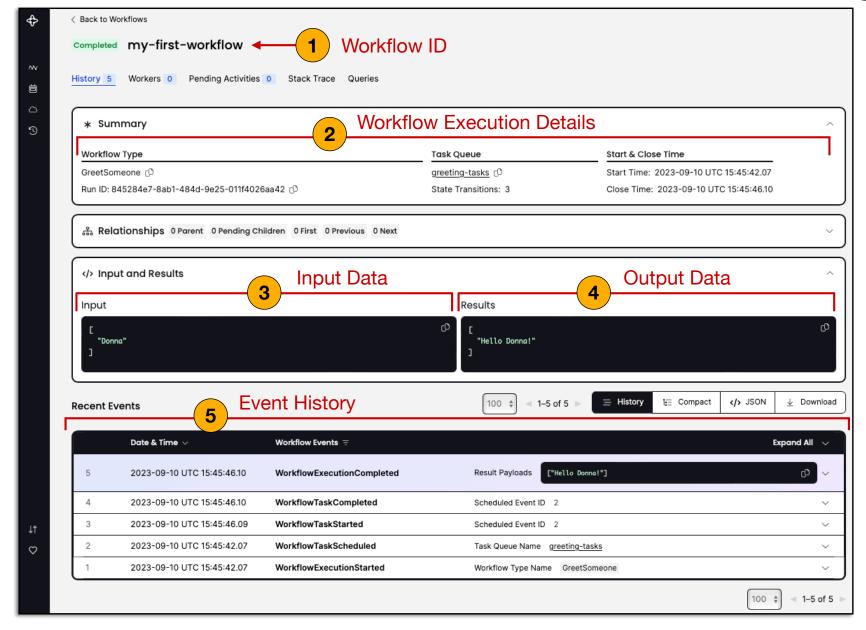
Viewing History from Web UI

- 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

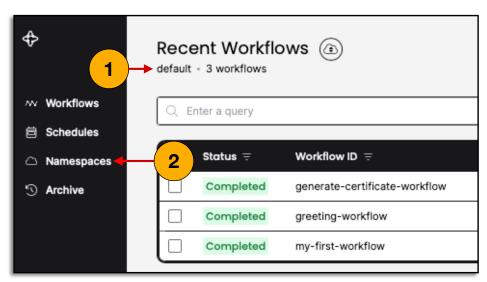


Web UI: Workflow Execution Detail Page



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



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 |
| 05 06 | Modifying an Existing Workflow Developing an Activity |
| | |
| 06 | Developing an Activity |

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 |
| 96 | 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

```
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

```
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

```
import (
    "go.temporal.io/sdk/workflow"
    "go.temporal.io/sdk/temporal" <</pre>
                                            Import this package from the SDK
    "time"
func GreetSomeone(ctx workflow.Context, name string) (string, error) {
    retrypolicy := &temporal.RetryPolicy {
      InitialInterval: 15 * time.Second, // first retry will occur after 15 seconds
                                    // double the delay after each retry
      BackoffCoefficient: 2.0,
      MaximumInterval: time.Second * 60, // up to a maximum delay of 60 seconds
                       100, // fail the Activity after 100 attempts
      MaximumAttempts:
    options := workflow.ActivityOptions {
        StartToCloseTimeout: time.Second * 5,
        RetryPolicy: retrypolicy, 
                                          Associate the policy with the Activity options
    ctx = workflow.WithActivityOptions(ctx, options)
    var result string
    err := workflow.ExecuteActivity(ctx, GreetInSpanish, name).Get(ctx, &result)
    // ... remainder of Workflow code would follow
```

Specify your policy values

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

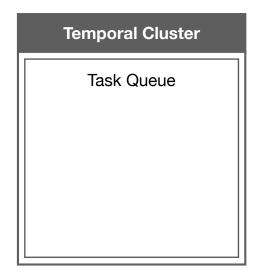
Temporal 101

| 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 |
| 98 | Understanding Workflow Execution |
| 09 | Conclusion |

Actors in this Workflow Execution Scenario



Executes the code

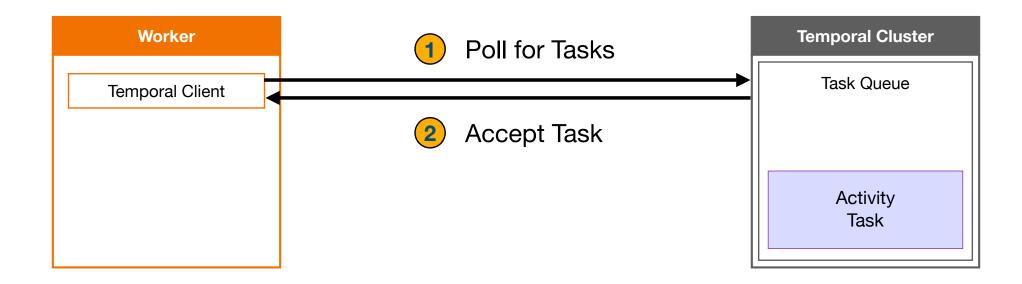


Orchestrates code execution



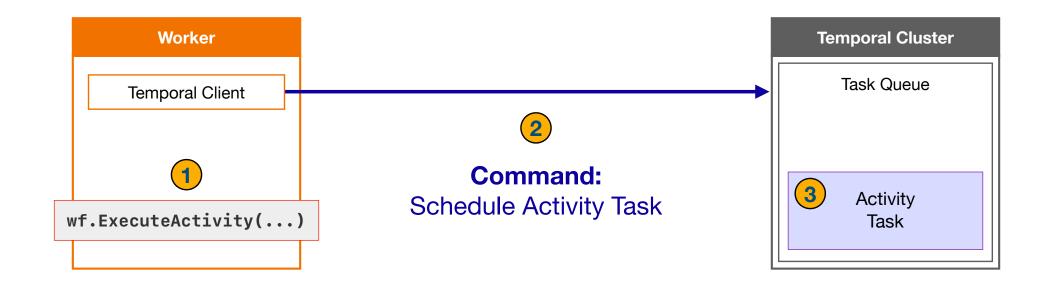
Requests code execution and retrieves the result

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

```
pushage fermeal // impart naturants united for browly

from Sentimentalistic connect connect, one string little, error)

return genius, error of "electronic particle particle
```

Workflow Definition

```
putbage formuli

inserting

"go. nagagari.in/subjection"

"go. nagagari.in/subjection

"go. nagagari.in/subjection

inserting in the subjection of the subje
```

Worker Initialization

```
pathop min

logati
framell "impuralled/neurrinos/fermell-mortfon/min
"po temporal-des/neurrinos/fermell-mortfon/min
"po temporal-des/neurrinos/fermell-mortfon/min
po temporal-des/neurrinos/fermell-mortfon/min
man morti (
mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortinos/fermell-mortin
```

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
                                                           This is just a utility function
   defer resp.Body.Close()
                                                           for calling the microservice
   body, err := ioutil.ReadAll(resp.Body)
                                                        and is not specific to Temporal
   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
```

```
package ferreal! // import statements mitted for bravity
from Contribugativities context, owns string (triting, error) (
return protein), never return proteins, prote
```

Workflow Definition

```
strong termil

"time"

"time"

"pure recognition of the foreign of
```

Worker Initialization

```
package main
import
framull 'temporalizi/secritam/faremil-mortfom/mai
'Temporalizi/secritam/faremil-mortfom/mai
'Temporalizi/secritam/faremil-mortfom/main
'Temporalizi/secritam/main
'Temporalizi/secritam/main
'Imporalizi/secritam/main
'Imporalizi/secri
```

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
```

```
package formula! // impurt naturants mitted for breatly
form Geneficial matter content, content, content of the content of the
```

Workflow Definition

```
pentage firmed:

import {
    "third "pentage firmed:
    "pentage firme
```

Worker Initialization

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 formula! // impurt visionest mainted for browity

them destributes indicate context context came stringe (intime, error) {
    return protein;
    return prot
```

Workflow Definition

```
puthup formuli
input |
"the"
"the"
"the proposal inforbinerflow"
"proposal inforbinerflow"
"proposal inforbinerflow"
"proposal inforbinerflow"
"form Continuermic to matrice Context, come string) (string, error) (
"therefore the proposal information of th
```

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)
```

Worker Process

Temporal Cluster

```
postage formed: // import without mainted for browity
four confidence of the confide
```

Workflow Definition

```
package formed]

interest that

"But representation of the content of the content
```

Worker Initialization

```
package min

lagest (

lag
```

```
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)
```

Worker Process

Temporal Client

Temporal Cluster

```
pushing terminal. // Empire interment mating for heavily
memoralization of the control of the co
```

Workflow Definition

```
prings formed]

import {
    "in.
    "
```

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)
```

Worker Process

Worker Entity

Temporal Client

Temporal Cluster

```
making transl1 // Super streament united for brevity
men ferrifording to content forces, using (string, error) (
greating, error in childrenical parties, using) (string, error) (
greating, error in childrenical parties) (string, error) (
error postering, error in childrenical parties) (string, error) (
error postering, error in childrenical parties) (
error postering, error in childrenical parties) (
error postering, error in childrenical parties) (
error postering, error in childrenical parties (
error postering) (error parties) (
error postering) (error parties) (
error postering) (error postering) postering) (e
```

Workflow Definition

```
pathogo fermentl
import (

"me.tomproxi.in/and/mentflor"

"me.tomproxi.in/and/mentflor"

"me.tomproxi.in/and/mentflor"

"startifications: time.former.

Startifications: time.former.

Startifications: time.former.

**startifications: time.former.

**startifications: time.former.

**startifications: time.former.

**startifications: time.former.

**startifications: time.former.

**restrian.**startifications:

**restrian.**startificat
```

Worker Initialization

```
pathops main

(sport |

(s
```

```
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)
```

Worker Process

Worker Entity

Temporal Client

Temporal Cluster

```
scales formula! // Super streament entire for horizon
men Sentedication context format, serving (string, string) and
graving, serving californizing-to-main-b-parking's, mean)
from Sentedication context format, serving (string, string) and
format serving serving serving-to-main-b-parking's, mean)
format serving serving serving serving
// string serving
// string
// string serving
// strin
```

Workflow Definition

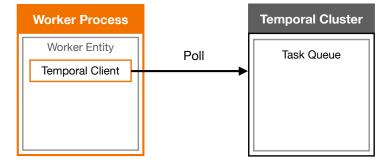
```
peckage formedi

inger

inger
```

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)
```



```
pastage formed1 // import statements mainted for trovity

from Contributation(control_control_men extract) (string_server) (
prints_pervice_mental_control_mental_control_mental_control_
prints_pervice_mental_control_mental_control_mental_control_
prints_pervice_mental_control_mental_control_mental_control_
prints_pervice_mental_control_mental_control_mental_control_
prints_pervice_mental_control_mental_control_mental_control_
prints_pervice_mental_control_mental_control_mental_control_
prints_pervice_mental_control_mental_control_
prints_pervice_mental_control_mental_control_
prints_pervice_mental_control_mental_control_
prints_pervice_mental_control_mental_control_
prints_pervice_mental_control_mental_control_
prints_pervice_mental_control_mental_control_
pervice_mental_control_mental_control_
pervice_mental_control_mental_control_
pervice_mental_control_mental_control_
pervice_mental_control_mental_control_
pervice_mental_control_pervice_mental_control_
pervice_mental_control_pervice_mental_control_
pervice_mental_control_pervice_mental_control_
pervice_mental_control_pervice_mental_control_pervice_mental_control_
pervice_mental_control_pervice_mental_control_pervice_mental_control_pervice_mental_control_pervice_mental_control_pervice_mental_control_pervice_mental_control_pervice_mental_control_pervice_mental_control_pervice_mental_control_pervice_mental_control_pervice_mental_control_pervice_mental_control_pervice_mental_control_pervice_mental_control_pervice_mental_control_pervice_mental_control_pervice_mental_control_pervice_mental_control_pervice_mental_control_pervice_mental_control_pervice_mental_control_pervice_mental_control_pervice_mental_control_pervice_mental_control_pervice_mental_control_pervice_mental_control_pervice_mental_control_pervice_mental_control_pervice_mental_control_pervice_mental_control_pervice_mental_control_pervice_mental_control_pervice_mental_control_pervice_mental_control_pervice_mental_control_pervice_mental_control_pervice_mental_control_pervice_mental_control_pervice_mental_
```

Workflow Definition

```
package frameal:
input:

"go.tompreal.in/adh/martine"

"go.tompreal.in/adh/martine"

"go.tompreal.in/adh/martine

Townthinessanders until no.denser , non extracy (string, error) (

Townthinessanders time.densed v. v.

two martinessanders time.densed v.
```

Worker Initialization

Launching from Command Line

```
\otimes \ominus \otimes
   $ tctl workflow run \
         --taskqueue greeting-tasks \
         --workflow_id greeting-workflow \
         --workflow_type GreetSomeone \
--input '"Tom"'
                                       Workflow
                                       Execution
                                       Request
                              Temporal Cluster
Worker Process
 Worker Entity
                    Poll
                                 Task Queue
 Temporal Client
```

```
protage formed] // import atthements mitted for involvy
free Contification(inc moters, Content, new string) (string, error) (
proteins, not - no distincted 'qui-question(), non)
}

for formedlindensimilet context. Content, new string) (string, error) (
protage of the context Content, new string) (string, error) (
protage protage of the context Content, new string) (string, error) (
protage protage of the content content of the content of
```

Workflow Definition

```
pentape fermed:

import {
    "in.
    "in.
```

Worker Initialization

```
package main
import [
import [
import ]
import [
```

Launching from Application Code

```
// ... 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{
               "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
                                          Temporal Cluster
          Worker Process
                                                                         Client Application
                                                             Workflow
                                                             Execution
            Worker Entity
                                             Task Queue
                               Poll
                                                             Request
            Temporal Client
                                                                           Temporal Client
```

```
pathogo formuli // Supert statement mitted for involve
foun Statisticalisations connect convex, mass strings (intrine, error) (
retine, particular, pa
```

Workflow Definition

Worker Initialization

```
pathage main

intered:

frameall "importality/exercises/frameall-most fine/malmine"

"m. importality/marcises/frameall-most fine/malmine"

"m. importality/marcises/
"m. importality/marcises/
"m. importality/marcises/
form small()

a "most section fine()

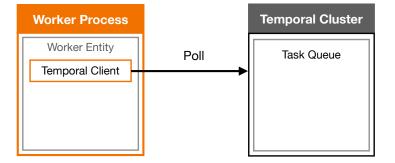
a "most section fine()

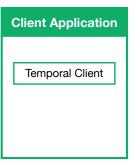
a "most section form small containment()

a "most section for form small
```

Event History

WorkflowExecutionStarted





```
package frameal // Super statements united for horsely
flow months of the control forces; an extract (strong error) (
return greating, are a "allestedis" packages (strong error) (
return greating, are
results (strong error) (
flow frame error) (
```

Workflow Definition

Worker Initialization

```
protops min
insert (
```

Worker Process Worker Entity Temporal Cluster Poll Temporal Cluster Workflow Task



Event History

WorkflowExecutionStarted

WorkflowTaskScheduled

```
rectange frameal // Journ tatements emitted for brewity

row finantifications content formers, now strain) strains, errors (
greeting, err : californic("pt-inpath-greating", nose)

return greating, err : californic("pt-inpath-greating", nose)

fone framealIndepanish(erc context.Context, now strain)

return greating, err

/ calling frames, normal-formersh()

return ()

return ()
```

Workflow Definition

```
package fermall
input (
"time"
"pa.regural.in/nd/mortlas"
"pa.regural.in/nd/mortlas"
"pa.regural.in/nd/mortlas"
"pa.regural.in/nd/mortlas"
"pa.regural.in/nd/mortlas"
"pa.regural.in/nd/mortlas"
"pa.regural.in/nd/mortlas"
"pa.regural.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mortlas.in/nd/mo
```

Worker Initialization

Worker Process Worker Entity Temporal Cluster Workflow Task Accept Task

Event History

WorkflowExecutionStarted

WorkflowTaskScheduled

WorkflowTaskStarted

Client Application

Temporal Client

```
package formula )/ (pages streament melting for horizo')

water (continuation content former, new tried) (string, error) (
generic, err (collisories'@st-emein-package, enter) (
generic, error) (collisories'@st-emein-package, enter) (
protein package)

former package (collisories) (
```

Workflow Definition

```
packep farmell
input (
"time"
"time"
"pa. (regresi.in/ch/netflor")
"pa. (regresi.in/ch/netflor")
"pa. (regresi.in/ch/netflor")
"pa. (regresi.in/ch/netflor")
"pa. (regresi.in/ch/netflor")
"pa. (regresi.in/ch/netflor")
"particles"
"part
```

Worker Initialization

```
package main
[agent 1]
[agent 1]
[agent 2]
[agent 3]
[agent 3]
[agent 4]
[agent 4]
[agent 4]
[agent 4]
[agent 4]
[agent 4]
[agent 5]
[agent 6]
[ag
```

```
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
}
```

Temporal Client

worker Process Worker Entity Poll Task Queue Wereturn "", err Client Application

Temporal Client

Workflow Task

Event History

WorkflowExecutionStarted

WorkflowTaskScheduled

WorkflowTaskStarted

```
package formall "/ impart turbunent mitted for bravity
those Genetideministic contact content, one string (tring, error) {
    retine, pressing, interfere benedicated to a content of the content of the
```

Workflow Definition

```
pockape formall

input(i

"go.temperal.in/in/hourtflar")

"go.temperal.in/in/hourtflar")

Fore EracEnseme(cit untflam.Contest, nom string) (utring, error) (

EracTtGCinerTiment: inh.Stonce * s.,

"tr = untflam.EracTtGcinerTiment; uption)

ver magnishDerating string
ver me untflam.EracMestivity(ris, PersitOgenia, nom).det(cts, AgaminGraving)

rise "", ver

ver quantiflam.EracMestivity(ris, FaramilloSgenia, nom).det(cts, AgaminGraving)

rise "",

ver quantiflam.EracMestivity(ris, FaramilloSgenia, nom).det(cts, AgaminGravani))

rise "",

ver quantiflam."

rise ""
```

Worker Initialization

```
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 Process Worker Entity Temporal Cluster Workflow Task Task



Event History

WorkflowExecutionStarted

WorkflowTaskScheduled

WorkflowTaskStarted

```
putpe formal // Squrt telement mitted for browly

Dought formal // Squrt telement mitted for browly

Form present of the state of the s
```

Workflow Definition

Worker Initialization

```
package main
input: [

'mpressin' 'temporalist/neuroisen/frameall-mostFam/mai
'mpressin' 'temporalist/neuroisen/frameall-mostFam/mai
'mpressin' (

'mpressin') (

'mpressin
```

```
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 Process Worker Entity Temporal Client Command: Schedule Activity Task Task Task Queue



Event History

WorkflowExecutionStarted

WorkflowTaskScheduled

WorkflowTaskStarted

WorkflowTaskCompleted

Workflow Definition

Worker Initialization

```
printing minimizer; impact in the property of the property of
```

Worker Process Worker Entity Temporal Cluster Poll Task Queue Activity Task



| WorkflowExecutionStarted |
|----------------------------------|
| WorkflowTaskScheduled |
| WorkflowTaskStarted |
| WorkflowTaskCompleted |
| ActivityTaskScheduled (Greeting) |

```
probag from 1 // input statements united for browity

"one proteins.context.context.context.context.go

greating.ger or calliferical/get-genich-geneting", near) (
proteins.geries, or calliferical/get-genich-geneting", near)

for from proteins.ger

for from the calliferical/get-genich-geneting (string, error) (
proteins.ger or sulforterical/genich-froms);

/ string proteins.ger

/ string
```

Workflow Definition

```
package farmali
import (
'liam'
'pa, respect, in/advant/lam'
'pa, respect to the resp
```

Worker Initialization

```
setting mini
[septing of the property of the p
```

Worker Process Worker Entity Temporal Client Activity Task Accept Task Temporal Cluster Task Queue

Client Application Temporal Client

| WorkflowExecutionStarte | d |
|-------------------------|------------|
| WorkflowTaskScheduled | |
| WorkflowTaskStarted | |
| WorkflowTaskCompleted | |
| ActivityTaskScheduled | (Greeting) |
| ActivityTaskStarted | (Greeting) |

```
pathage formula! // impurt naturants united for heavity

how Statistical middlets connect connect cone tring listing, error!

pathage pathage control of the control management of the control of the con
```

Workflow Definition

Worker Initialization

```
package main
insuri_
framell 'temprellEf/avertises/framell-entfin/ml
'pp. temprell-info/hitest'
'pp. temprell-info/hitest'
'pp. temprell-info/hitest'
'pp. temprell-info/hitest'
'pp. temprell-info/hitest'
'pp. temprell-info/hitest'
'tem risel'
'te
```

```
// 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
```

Worker Process Worker Entity Temporal Client Activity Task Task Temporal Cluster

Client Application Temporal Client

| WorkflowExecutionStarted | d |
|--------------------------|------------|
| WorkflowTaskScheduled | |
| WorkflowTaskStarted | |
| WorkflowTaskCompleted | |
| ActivityTaskScheduled | (Greeting) |
| ActivityTaskStarted | (Greeting) |

```
package formunil // impurt attements united for hrowity

Anno Emethodymicalitic connect Context, mess string) (train, error) (
prints, general)

From general context, mess string (train, error) (
protein, general)

From general context Context, mes string (train, error) (
protein, general context Context, mes string) (train, error) (
protein, general context Context, mes string) (train, error) (
protein, general context Context, mess string) (train, error) (
protein, general context, general co
```

Workflow Definition

```
peckage fermatil
inger()

"go.tomprat.in/ab/hometflar")

"for Considerative swotflam.Omitet, non string) (string, error) (

StartScientiment: sim.Scient * s.,

Its variation.Ministriptime(rs, spring)

err swotflam.Scientiment.ins.Scient * s.,

err swotflam.Scientiment.ins.Scientiment.

err swotflam.Scientiment.

err
```

Worker Initialization

Translation

Access microservice

and request greeting

```
package main
input: [

'mpressin' 'temporalist/neuroisen/frameall-mostFam/mai
'mpressin' 'temporalist/neuroisen/frameall-mostFam/mai
'mpressin' (

'mpressin') (

'mpressin
```

```
// 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
```

Worker Process Worker Entity Temporal Cluster Poll Temporal Client Activity Task

Client Application Temporal Client

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

```
package formula! // impurt statements united for browity

but Small statement of the control of the statement of the control o
```

Workflow Definition

```
package fermedi
input (

"ge.tegenzi.in/inferentize")

"ge.tegenzi.in/inferentize"

"fermediamentize artifize.Contat, nom string) (string, error) (

"Startifize.inferent time.Second * 2,

"the "entitin with behaviory in the string of the st
```

Worker Initialization

Translation

Translation service

responds with greeting

```
prince minima to the control of the
```

```
// 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
```

Worker Process Worker Entity Temporal Client Activity Task Task Temporal Cluster

Client Application Temporal Client

| WorkflowExecutionStarte | d |
|-------------------------|------------|
| WorkflowTaskScheduled | |
| WorkflowTaskStarted | |
| WorkflowTaskCompleted | |
| ActivityTaskScheduled | (Greeting) |
| ActivityTaskStarted | (Greeting) |

```
package formula )/ (pages telements militad for twenty me, gravity (string, error) ( previous formula ) ( previous
```

Workflow Definition

```
pendage formed!

imput (
    "title"
    "p.,tapequati.in/sh/mortflat"
    "p.,tapequati.in/sh/mortflat"
    "p.,tapequati.in/sh/mortflat"
    "p.,tapequati.in/sh/mortflat"
    "p. tapequati.in/sh/mortflat"
    "p. tapequati.in/sh/mortflat"
    "p. tapequati.in/sh/mortflat"
    "p. tapequati.in/sh/mortflat"
    "p. tapequati.in/sh/mortflat"
    "p. tapequati.in/sh/mortflat
    "p. tapequati.in
```

Worker Initialization

```
making min

[signt (

[sig
```

```
// 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
```

Worker Process Worker Entity Temporal Client Notify Activity Task Complete Task Queue

Client Application Temporal Client

| WorkflowExecutionStarted | t |
|--------------------------|------------|
| WorkflowTaskScheduled | |
| WorkflowTaskStarted | |
| WorkflowTaskCompleted | |
| ActivityTaskScheduled | (Greeting) |
| ActivityTaskStarted | (Greeting) |
| ActivityTaskCompleted | (Greeting) |

```
prology from 1 // import streament united for horsely

"more greated_passing, comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestion_comestio
```

Workflow Definition

```
pathops fermall
import (
"tiles"
"pi.remyszi.tivish/pentflam"
"pi.remyszi.tivish/pentflam"
"pi.remyszi.tivish/pentflam"
"pi.remyszi.tivish/pentflam"
(prises v = martine Acitivish/penter)
(prises v = martine Acitivish/pente
```

Worker Initialization

```
patters main
figure 1
for the control of the contro
```

Worker Process Worker Entity Temporal Cluster Poll Temporal Client Workflow Task



| WorkflowExecutionStarted | I |
|--------------------------|------------|
| WorkflowTaskScheduled | |
| WorkflowTaskStarted | |
| WorkflowTaskCompleted | |
| ActivityTaskScheduled | (Greeting) |
| ActivityTaskStarted | (Greeting) |
| ActivityTaskCompleted | (Greeting) |
| WorkflowTaskScheduled | |

```
package formula! // injust statements mainted for horizing forms of the control o
```

Workflow Definition

```
perhaps format!

input:

'gu, request.in/nu/nurfiles'

'gu, request.in/nu/nurfiles'

'gu, request.in/nu/nurfiles'

'gu, request.in/nurfiles contact, rest string (string, error) (

outstands.in/nurfiles request.in/nurfiles (

information transferring to the contact of a,

cts - wearfiles (distribution) transferring to a,

cts - wearfiles (distribution) transferring to a,

cts - wearfiles (distribution) transferring tra
```

Worker Initialization

```
parting main layers of the properties of the pro
```

Worker Process Worker Entity Temporal Client Workflow Task Accept Task Temporal Cluster Task Queue

Client Application Temporal Client

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

```
pacings formed] // Sports teatments emitted for brewity
and contribution(content_Content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_content_conte
```

Workflow Definition

```
packape formuli
input (
"time"
"time"
"time"
"time"
"put (
"time"
"time"
"time"
"time"
"time"
"time"
"time"
"time"
"time"
"time (
"time"
"time
```

Worker Initialization

```
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 Process Worker Entity Temporal Cluster Task Queue Workflow Task

Client Application Temporal Client

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

```
pacings formed] // Sports testiments united for brewly
and contribution(c) control (control, control, control,
grants, err : enlikeries('met-speaks-paraling, rene) (
grants, err : enlikeries('met-speaks-paraling, rene)
form pacing, err : enlikeries('met-speaks-paraling, rene)
form pacing ('met-speaks-paraling, rene)
form pacing ('met-speaks-paraling, rene)
err : enlikeries('met-speaks-paraling, rene)
err : enlikeries('met-speaks-paraling, rene)
form pacing, err : enlikeries('
```

Workflow Definition

```
protop formall
input {
    "time" |
    "time" |
    "time" |
    "time |
    "time" |
    "time |
```

Worker Initialization

```
package main
[agent 1]
[agent 1]
[agent 2]
[agent 3]
[agent 3]
[agent 4]
[agent 4]
[agent 4]
[agent 4]
[agent 4]
[agent 4]
[agent 5]
[agent 6]
[ag
```

```
// ... 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
}
```

Worker Process Worker Entity Temporal Client Command: Schedule Activity Task Task Task Queue

Client Application Temporal Client

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

```
peckage frameal // input statements unitted for horizing frameal // input statements unitted for horizing, error) (
processing, err : exilterizating-to-penint-greating-trans) 
return processing, err
peckage frameal frameal // input statements //
```

Workflow Definition

Worker Initialization

```
setups min
import |
i
```

Worker Process Worker Entity Temporal Cluster Poll Temporal Client Activity Task

Client Application Temporal Client

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

```
pathage formeal // impact statements mitted for browity

Ame SurethSquesimites notated_context, mose string (tring, error) (
    return practic, most received_context, mose string (tring, error) (
    return practic, most received_context, most return grating, error)

for formed_context.context_denset, most return (tring, error) (
    practing, orr = collectived_practic_formed);

/ willife formed_context_denset, most return (tring, error) (
    vall formed_context_denset, most return (
    vall formed_context_denset, most formed
```

Workflow Definition

```
package farmed]
input {
    "time"    "time"
```

Worker Initialization

```
package main

fagers (

"mineral temperalEff/section/framell-mortfon/mil

"mineral temperalEff/section/framell-mortfon/mil

"mineral temperalEff/section

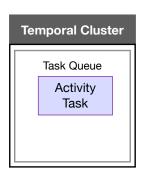
"mineralEff/section/mile

for mineralEff/section/mile

for minera
```

What happens if the Worker crashes?







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

```
package formula! // injust statements mainted for horizing forms of the control o
```

Workflow Definition

Worker Initialization

```
process min

from 1 to process the first to the first to
```

Worker Process Worker Entity Temporal Cluster Poll Task Queue Activity Task Accept Task

Client Application Temporal Client

| WorkflowExecutionStarte | d |
|-------------------------|------------|
| WorkflowTaskScheduled | |
| WorkflowTaskStarted | |
| WorkflowTaskCompleted | |
| ActivityTaskScheduled | (Greeting) |
| ActivityTaskStarted | (Greeting) |
| ActivityTaskCompleted | (Greeting) |
| WorkflowTaskScheduled | |
| WorkflowTaskStarted | |
| WorkflowTaskCompleted | |
| ActivityTaskScheduled | (Farewell) |
| ActivityTaskStarted | (Farewell) |

```
package formed] // (more statements mainted for twenty)

more contained that the content of the
```

Workflow Definition

```
penhape formed |
import {
    "yp.temporal.in/ads/montflee"
    "yp.temporal.in/ads/montflee"
    "yp.temporal.in/ads/montflee"
    "yp.temporal.in/ads/montflee"
    "yp.temporal.in/ads/montflee
    "yp.temporal.in/ads/montflee
```

Worker Initialization

```
// 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
```

Worker Process Worker Entity Temporal Client Activity Task Task Temporal Cluster

Client Application Temporal Client

| WorkflowExecutionStarte | ed |
|-------------------------|------------|
| WorkflowTaskScheduled | |
| WorkflowTaskStarted | |
| WorkflowTaskCompleted | |
| ActivityTaskScheduled | (Greeting) |
| ActivityTaskStarted | (Greeting) |
| ActivityTaskCompleted | (Greeting) |
| WorkflowTaskScheduled | |
| WorkflowTaskStarted | |
| WorkflowTaskCompleted | |
| ActivityTaskScheduled | (Farewell) |
| ActivityTaskStarted | (Farewell) |
| | |

```
package formula! // journt trainments mitted for browity

but more and the second content of the content of the
```

Workflow Definition

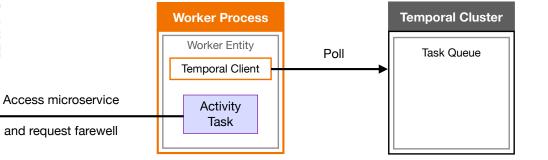
```
princips formula

input (
    "then"
    "then"
    "then"
    "then"
    "then"
    "then (imputaling the house from a tring) (string, error) (
    grains : marking details (bottom, man tring) (string, error) (
    grains : marking details (string)
    production of the string of the string (string)
    "then are a string to tring (string)
    "then are a string to the string (string)
    "the string to the string to the string (string)
    "the string to the string to the string (string)
    "the string to the s
```

Worker Initialization

Translation

```
// 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
```



Client Application Temporal Client

| WorkflowExecutionStarte | d |
|-------------------------|------------|
| WorkflowTaskScheduled | |
| WorkflowTaskStarted | |
| WorkflowTaskCompleted | |
| ActivityTaskScheduled | (Greeting) |
| ActivityTaskStarted | (Greeting) |
| ActivityTaskCompleted | (Greeting) |
| WorkflowTaskScheduled | |
| WorkflowTaskStarted | |
| WorkflowTaskCompleted | |
| ActivityTaskScheduled | (Farewell) |
| ActivityTaskStarted | (Farewell) |
| | |

```
gackage fermail // import statements mixted for brevity

from Genetificationistics monetal context, ness string (string, error) (
return protein); new financial context, ness string (string, error) (
protein, protein, error collisionist's question, error collisionist's question, error collisionist's question, error collisionist's question, formation protein, error collisionist's question, error collisionist's question, error collisionist's question, error (
protein protein);

("Willify months for mealing calls to the microscopical collisionist's collisionist's question error collisionist's questionist's collisionist's collisioni
```

Workflow Definition

```
probability formula

import (

"tlan"

"p. negrous (cit swell-n.cotte, one string) (tring, error) (

more foresteement in definition of the cotte, one string) (tring, error) (

more foresteement in decours, one string) (tring, error) (

more foresteement in decours of the cotte, one string) (tring, error) (

cut swell-n.cotte, one of the cotte, one of
```

Worker Initialization

Service Unavailable

```
package min

input (

frame) 'topopullif/mercian/framed)-markin/malmin'

'go.teppal.in/malmin'

'go.teppal.in/malmin'

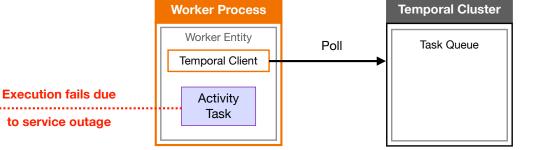
'go.teppal.in/malmin'

for.spin() (

c.spr = class.disidinim-dprime())

input i
```

```
// 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
                                                                                    Error
   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
```



Client Application Temporal Client

| WorkflowExecutionStarte | d |
|-------------------------|------------|
| WorkflowTaskScheduled | |
| WorkflowTaskStarted | |
| WorkflowTaskCompleted | |
| ActivityTaskScheduled | (Greeting) |
| ActivityTaskStarted | (Greeting) |
| ActivityTaskCompleted | (Greeting) |
| WorkflowTaskScheduled | |
| WorkflowTaskStarted | |
| WorkflowTaskCompleted | |
| ActivityTaskScheduled | (Farewell) |
| ActivityTaskStarted | (Farewell) |
| | |

```
primage fermal) // Depart abstances united for browing
from Constitutional Content of the Conten
```

Workflow Definition

```
probage formedl

input.[
"input.].in/ab/martine"

"in. imput.].in/ab/martine"

"in. imput.].in/ab/martine"

"in. imput.].in/ab/martine"

Surticular/imput.].in/ab/martine"

Surticular/imput.].in/ab/martine

Surticular/imput.].in/ab/martine

Surticular/imput.].in/ab/martine

ver special/imput.

ver special/
```

Worker Initialization

Translation

```
primage main
imput [
i
```

```
// 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
                                                           Activity is invoked
   return greeting, nil
                                                            again during retry
// 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
```

Worker Process Worker Entity Temporal Client Access microservice Activity Task Activity Task Temporal Cluster

Client Application Temporal Client

| WorkflowExecutionStarte | d |
|-------------------------|------------|
| WorkflowTaskScheduled | |
| WorkflowTaskStarted | |
| WorkflowTaskCompleted | |
| ActivityTaskScheduled | (Greeting) |
| ActivityTaskStarted | (Greeting) |
| ActivityTaskCompleted | (Greeting) |
| WorkflowTaskScheduled | |
| WorkflowTaskStarted | |
| WorkflowTaskCompleted | |
| ActivityTaskScheduled | (Farewell) |
| ActivityTaskStarted | (Farewell) |
| | |

```
pacings formed] // Sports testiments united for brewly
and contribution(c) control (control, control, control,
grants, err : enlikeries('met-speaks-paraling, rene) (
grants, err : enlikeries('met-speaks-paraling, rene)
form pacing, err : enlikeries('met-speaks-paraling, rene)
form pacing ('met-speaks-paraling, rene)
form pacing ('met-speaks-paraling, rene)
err : enlikeries('met-speaks-paraling, rene)
err : enlikeries('met-speaks-paraling, rene)
form pacing, err : enlikeries('
```

Workflow Definition

Worker Initialization

Translation

Translation service

responds with farewell

```
// 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
```

Worker Process Worker Entity Temporal Cluster Poll Temporal Client Activity Task

Client Application Temporal Client

| WorkflowExecutionStarte | d |
|-------------------------|------------|
| WorkflowTaskScheduled | |
| WorkflowTaskStarted | |
| WorkflowTaskCompleted | |
| ActivityTaskScheduled | (Greeting) |
| ActivityTaskStarted | (Greeting) |
| ActivityTaskCompleted | (Greeting) |
| WorkflowTaskScheduled | |
| WorkflowTaskStarted | |
| WorkflowTaskCompleted | |
| ActivityTaskScheduled | (Farewell) |
| ActivityTaskStarted | (Farewell) |
| | |

```
package frameail // import attaments united for knowly

from Smartheness of the control of the c
```

Workflow Definition

```
package formed)
ingent
```

Worker Initialization

```
// 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
```

Worker Process Worker Entity Temporal Client Notify Activity Task Complete Temporal Cluster Task Queue

Client Application Temporal Client

| WorkflowExecutionStarte | d |
|-------------------------|------------|
| WorkflowTaskScheduled | |
| WorkflowTaskStarted | |
| WorkflowTaskCompleted | |
| ActivityTaskScheduled | (Greeting) |
| ActivityTaskStarted | (Greeting) |
| ActivityTaskCompleted | (Greeting) |
| WorkflowTaskScheduled | |
| WorkflowTaskStarted | |
| WorkflowTaskCompleted | |
| ActivityTaskScheduled | (Farewell) |
| ActivityTaskStarted | (Farewell) |
| ActivityTaskCompleted | (Farewell) |
| | |

```
package formula // depart statements motited for haveiny
non-security and the security control of the security of the security
```

Workflow Definition

```
penhaps formatil
input

(input

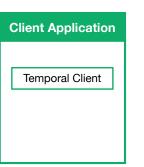
(input
```

Worker Initialization

```
sector make
import [

'approximate and important procession of the control of the
```

Worker Process Worker Entity Temporal Cluster Task Queue Workflow Task



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

```
package frameal. // input statements unitted for browity
free Grantingoldrice constitutions, come string (triun, serse) (
granting, err or callierscalega-speaks-granting, ranse)
reines granting, err
silver, errespondent, content, content
```

Workflow Definition

```
peskap framell

import (

"title "

"p. tapport.in/sh/hortflar"

"p. tappo
```

Worker Initialization

```
process min

from 1 to process the first to the first to
```

Worker Process Worker Entity Temporal Cluster Poll Task Queue Workflow Task Accept Task

Client Application Temporal Client

| WorkflowExecutionStarte | d |
|-------------------------|------------|
| WorkflowTaskScheduled | |
| WorkflowTaskStarted | |
| WorkflowTaskCompleted | |
| ActivityTaskScheduled | (Greeting) |
| ActivityTaskStarted | (Greeting) |
| ActivityTaskCompleted | (Greeting) |
| WorkflowTaskScheduled | |
| WorkflowTaskStarted | |
| WorkflowTaskCompleted | |
| ActivityTaskScheduled | (Farewell) |
| ActivityTaskStarted | (Farewell) |
| ActivityTaskCompleted | (Farewell) |
| WorkflowTaskScheduled | |
| WorkflowTaskStarted | |

```
resp, err := http.Get(url)
if err != nil (
    return **, err
)
defer resp.Body.Close()
body, err := ioutil.ReadAll(resp.Sody)
if err != nil (
    return **, err
}
```

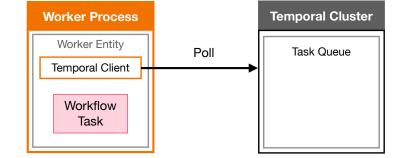
Workflow Definition

```
package faremell
import (
"time"
     return helloGoodbye, nil
```

Worker Initialization

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

```
// ... 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
```





| WorkflowExecutionStarte | ed |
|-------------------------|------------|
| WorkflowTaskScheduled | |
| WorkflowTaskStarted | |
| WorkflowTaskCompleted | |
| ActivityTaskScheduled | (Greeting) |
| ActivityTaskStarted | (Greeting) |
| ActivityTaskCompleted | (Greeting) |
| WorkflowTaskScheduled | |
| WorkflowTaskStarted | |
| WorkflowTaskCompleted | |
| ActivityTaskScheduled | (Farewell) |
| ActivityTaskStarted | (Farewell) |
| ActivityTaskCompleted | (Farewell) |
| WorkflowTaskScheduled | |
| WorkflowTaskStarted | |
| | |

```
sating formula // (paper thisments mitted for hereity
mem. Constitutionalist constant Centur, most time) intring, error (
proving, err : callistrative ("met-most-hereity), mems)
from proving and constant constant constant constant constant
from formula ("mems, constant con
```

Workflow Definition

```
stateps formulal
input (
"time"
"ps. regress.in/nch/nortflor"
Then Sentimenencies working.Context, non string) (string, error) (
ordinal to working.Context, non string) (string, error) (
ordinal to working.Context, non-string) (string, error) (
ordinal to marking.context, non-string) (
ordinal to marking.context, non-string)
ordinal to marking.context, non-string, non-string,
```

Worker Initialization

```
package main
impact [
frame] 'temporallEf/nermines/frame]-outflow/mi
'mp. temporallef/nermines/frame]
'mp. temporallef/nermines/frame]
'mp. temporallef/nermines/mi
'mp. temporallef/nermines/
```

```
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 Process Worker Entity Temporal Client Task Queue

Client Application Temporal Client

| WorkflowExecutionStarted | t |
|--------------------------|------------|
| WorkflowTaskScheduled | |
| WorkflowTaskStarted | |
| WorkflowTaskCompleted | |
| ActivityTaskScheduled | (Greeting) |
| ActivityTaskStarted | (Greeting) |
| ActivityTaskCompleted | (Greeting) |
| WorkflowTaskScheduled | |
| WorkflowTaskStarted | |
| WorkflowTaskCompleted | |
| ActivityTaskScheduled | (Farewell) |
| ActivityTaskStarted | (Farewell) |
| ActivityTaskCompleted | (Farewell) |
| WorkflowTaskScheduled | |
| WorkflowTaskStarted | |
| WorkflowTaskCompleted | |
| | |

```
product recently // Equat statument matter for barety consideration of the control of the contro
```

Workflow Definition

Worker Initialization

```
package main

imput

formall 'imputallE/corrinos/farcedil-quefica/main

'put temporal.in/she/placet

formall 'imputallE/corrinos/farcedil-quefica/main

'put temporal.in/she/placet

formall (

imputallE/corrinos/farcedil-quefica/

putallE/corrinos/farcedil-quefica/

putallE/corrinos/farcedil-quefica/

imputallE/corrinos/farcedil-quefica/

imputallE/corrinos/farcedil-quefica/

imputallE/corrinos/farcedil-quefica/

imputallE/corrinos/farcedil-quefica/

imputallE/corrinos/farcedil-quefica/

imputallE/corrinos/farcedil-quefica/

imputallE/corrinos/farcedil-quefica/

imputallE/corrinos/farcedil-quefica/

imputallE/corrinos/farcedil-quefica/

jacconstruction/farcedil-quefica/

ja
```

Temporal Cluster

Task Queue

Request result

Worker Process

Worker Entity

Temporal Client

Poll

Client Application

Temporal Client

| WorkflowExecutionStarted | |
|---------------------------|------------|
| WorkflowTaskScheduled | |
| WorkflowTaskStarted | |
| WorkflowTaskCompleted | |
| ActivityTaskScheduled | (Greeting) |
| ActivityTaskStarted | (Greeting) |
| ActivityTaskCompleted | (Greeting) |
| WorkflowTaskScheduled | |
| WorkflowTaskStarted | |
| WorkflowTaskCompleted | |
| ActivityTaskScheduled | (Farewell) |
| ActivityTaskStarted | (Farewell) |
| ActivityTaskCompleted | (Farewell) |
| WorkflowTaskScheduled | |
| WorkflowTaskStarted | |
| WorkflowTaskCompleted | |
| WorkflowExecutionComplete | ed |

```
package fermall // import statements united for browity
fone Gentidopatholics context. Context. one string) (triting, error) (
return percents;
fore forentialization of the context. one string) (triting, error) (
genting, error - collisionical quintum, context of triting errors;
fone relationship of the collisionical quintum, context of triting particle, err
// unitily foreither enableg collis to the discussrians
fone californication string, one string) (triting, error) (
string, error of the collisionical quintum, error)
error, error of this, desired
error, error, triting, unit.desprinceplanes)

resp. error of this, desired
error, error, desprinceplanes
if error, error, desprinceplanes
if error, error, desprinceplanes
if error, error, desprinceplanes

treadstation of error, desprinceplanes

resp. error, desprinceplanes

resp. error, desprinceplanes

treadstation of error, desprinceplanes

return treadstation, etc.
```

Workflow Definition

Worker Initialization

```
primage main
imput [
i
```

The End

```
// ... this is code within your own application (for example, a web application, mobile app, etc.)
options := client.StartWorkflowOptions{
               "greeting-workflow",
    ID:
   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
                                       Temporal Cluster
                                                                      Client Application
         Worker Process
```

Task Queue

Provide result

Temporal Client

Worker Entity

Temporal Client

Poll

| WorkflowExecutionStarted | |
|--------------------------|------------|
| WorkflowTaskScheduled | |
| WorkflowTaskStarted | |
| WorkflowTaskCompleted | |
| ActivityTaskScheduled | (Greeting) |
| ActivityTaskStarted | (Greeting) |
| ActivityTaskCompleted | (Greeting) |
| WorkflowTaskScheduled | |
| WorkflowTaskStarted | |
| WorkflowTaskCompleted | |
| ActivityTaskScheduled | (Farewell) |
| ActivityTaskStarted | (Farewell) |
| ActivityTaskCompleted | (Farewell) |
| WorkflowTaskScheduled | |
| WorkflowTaskStarted | |
| WorkflowTaskCompleted | |
| WorkflowExecutionComplet | ed |

Temporal 101

| 05 | Modifying an Existing Workflow |
|----|------------------------------------|
| 04 | Viewing Workflow Execution History |
| 04 | Viewing Workflow Execution History |
| 04 | Viewing Workflow Execution History |
| | |
| 03 | Executing a Workflow |
| | |
| 02 | Developing a Workflow |
| 01 | What is Temporal? |
| 00 | About this Workshop |

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