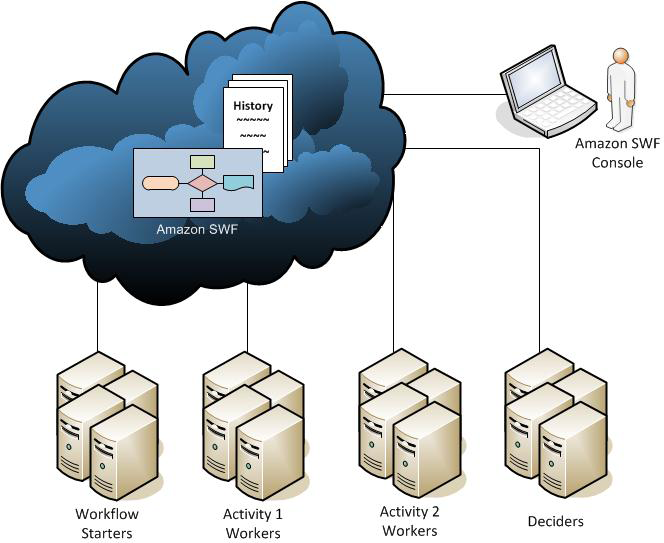
C# Hello world! Amazon Simple Workflow (SWF)

Sivaprasad Padisetty (May 2014)

*The Amazon Simple Workflow Service (Amazon SWF) makes it easy to build scalable distributed applications that coordinate work across distributed components. In this blog I will share a simple C# program to illustrate basic concepts.*

# Concepts

The following diagram shows the Amazon SWF architecture, including Amazon SWF and its actors



## Workflow

The fundamental concept in Amazon SWF is the workflow. A workflow is a set of activities that carry out some objective, together with logic that coordinates the activities. For example, a workflow could receive a customer order and take whatever actions are necessary to fulfill it. Each workflow runs in an AWS resource called a domain, which controls the workflow's scope. An AWS account can have multiple domains, each of which can contain multiple workflows, but workflows in different domains cannot interact.

## Workflow Starters

A workflow starter is any application that can initiate workflow executions.

## Activity Workers

An activity worker is a process or thread that performs the activity tasks that are part of the workflow. (i.e.) Activity task is one step in the workflow. To use an activity task, you must register it using either the Amazon SWF console or the RegisterActivityType action.

Each activity worker polls Amazon SWF for new tasks that are appropriate for that activity worker to perform; certain tasks can be performed only by certain activity workers. After receiving a task, the activity worker processes the task to completion and then reports to Amazon SWF that the task was completed and provides the result. The activity worker then polls for a new task. The activity workers associated with a workflow execution continue in this way, processing tasks until the workflow execution itself is complete. Activity workers can run in AWS or in your datacenter behind the firewall.

## Deciders

A decider is an implementation of a workflow's coordination logic. Deciders control the flow of activity tasks in a workflow execution. Whenever a change occurs during a workflow execution, such as the completion of an activity task, Amazon SWF creates a decision task that contains the workflow history up to that point in time and assigns the task to a decider. When the decider receives the decision task from Amazon SWF, it analyzes the workflow execution history to determine the next appropriate steps in the workflow execution. The decider communicates these steps back to Amazon SWF using decisions. A decision is an Amazon SWF data type that can represent various next actions.

# Workflow Execution

1. Write activity workers that implement the processing steps in your workflow.
2. Write a decider to implement the coordination logic of your workflow.
3. Register your activities and workflow with Amazon SWF. You can do this step programmatically or by using the AWS Management Console.
4. Start your activity workers and decider. These actors can run on any computing device that can access an Amazon SWF endpoint. For example, you could use compute instances in the cloud, such as Amazon Elastic Compute Cloud (Amazon EC2); servers in your data center; Once started, the decider and activity workers should start polling Amazon SWF for tasks.
5. Start one or more executions of your workflow. Executions can be initiated either programmatically or via the AWS Management Console. Each execution runs independently and you can provide each with its own set of input data. When an execution is started, Amazon SWF schedules the initial decision task. In response, your decider begins generating decisions which initiate activity tasks. Execution continues until your decider makes a decision to close the execution.
6. View workflow executions using the AWS Management Console. You can filter and view complete details of running as well as completed executions. For example, you can select an open execution to see which tasks have completed and what their results were.

# App.config

App.config defines what profile and region to use. In the example below it uses the default profile that PowerShell uses. Important: Please don’t save your accesskey & secretkey, easy to lose when you publish the file.

<?xml version="1.0"?>

<configuration>

<appSettings>

<add key="AWSProfileName" value="AWS PS Default"/>

<add key="AWSRegion" value="us-east-1" />

</appSettings>

<startup>

<supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5" />

</startup>

</configuration>

# main

Code below shows the using statements and main. The main does the setup first, then launches workers, deciders and starts the workflow. Activity1A and Activity1B are serviced by one set of workers (they share the same tasklist name Activity1). Activity2 is serviced by another set of workers.

using System;

using System.Collections.Generic;

using System.Threading.Tasks;

using System.Linq;

using Amazon;

using Amazon.SimpleWorkflow;

using Amazon.SimpleWorkflow.Model;

static string domainName = "HelloWorldDomain";

static IAmazonSimpleWorkflow swfClient =

AWSClientFactory.CreateAmazonSimpleWorkflowClient();

public static void Main(string[] args)

{

string workflowName = "HelloWorld Workflow";

// Setup

RegisterDomain();

RegisterActivity("Activity1A", "Activity1");

RegisterActivity("Activity1B", "Activity1");

RegisterActivity("Activity2", "Activity2");

RegisterWorkflow(workflowName);

// Launch workers to service Activity1A and Activity1B

// This is acheived by sharing same tasklist name (i.e.) "Activity1"

Task.Run(() => Worker("Activity1"));

Task.Run(() => Worker("Activity1"));

// Launch Workers for Activity2

Task.Run(() => Worker("Activity2"));

Task.Run(() => Worker("Activity2"));

// Start the Deciders, which defines the structure/flow of Workflow

Task.Run(() => Decider());

//Start the workflow

Task.Run(() => StartWorkflow(workflowName));

Console.Read();

}

# Register Workflow and Activities

static void RegisterDomain()

{

// Register if the domain is not already registered.

var listDomainRequest = new ListDomainsRequest()

{

RegistrationStatus = RegistrationStatus.REGISTERED

};

if (swfClient.ListDomains(listDomainRequest).DomainInfos.Infos.FirstOrDefault(

x => x.Name == domainName) == null)

{

RegisterDomainRequest request = new RegisterDomainRequest()

{

Name = domainName,

Description = "Hello World Demo",

WorkflowExecutionRetentionPeriodInDays = "1"

};

Console.WriteLine("Setup: Created Domain - " + domainName);

swfClient.RegisterDomain(request);

}

}

static void RegisterActivity (string name, string tasklistName)

{

// Register activities if it is not already registered

var listActivityRequest = new ListActivityTypesRequest()

{

Domain = domainName,

Name = name,

RegistrationStatus = RegistrationStatus.REGISTERED

};

if (swfClient.ListActivityTypes(listActivityRequest).ActivityTypeInfos.TypeInfos.FirstOrDefault(

x => x.ActivityType.Version == "2.0") == null)

{

RegisterActivityTypeRequest request = new RegisterActivityTypeRequest()

{

Name = name,

Domain = domainName,

Description = "Hello World Activities",

Version = "2.0",

DefaultTaskList = new TaskList() { Name = tasklistName },//Worker poll based on this

DefaultTaskScheduleToCloseTimeout = "300",

DefaultTaskScheduleToStartTimeout = "150",

DefaultTaskStartToCloseTimeout = "450",

DefaultTaskHeartbeatTimeout = "NONE",

};

swfClient.RegisterActivityType(request);

Console.WriteLine("Setup: Created Activity Name - " + request.Name);

}

}

static void RegisterWorkflow(string name)

{

// Register workflow type if not already registered

var listWorkflowRequest = new ListWorkflowTypesRequest()

{

Name = name,

Domain = domainName,

RegistrationStatus = RegistrationStatus.REGISTERED

};

if (swfClient.ListWorkflowTypes(listWorkflowRequest).WorkflowTypeInfos.TypeInfos.FirstOrDefault (

x => x.WorkflowType.Version == "2.0") == null)

{

RegisterWorkflowTypeRequest request = new RegisterWorkflowTypeRequest()

{

DefaultChildPolicy = ChildPolicy.TERMINATE,

DefaultExecutionStartToCloseTimeout = "300",

DefaultTaskList = new TaskList()

{

Name = "HelloWorld" // Decider need to poll for this task

},

DefaultTaskStartToCloseTimeout = "150",

Domain = domainName,

Name = name,

Version = "2.0"

};

swfClient.RegisterWorkflowType(request);

Console.WriteLine("Setup: Registerd Workflow Name - " + request.Name);

}

}

# StartWorkflow

static void StartWorkflow (string name)

{

IAmazonSimpleWorkflow swfClient = AWSClientFactory.CreateAmazonSimpleWorkflowClient();

string workflowID = "Hello World WorkflowID - " + DateTime.Now.Ticks.ToString();

swfClient.StartWorkflowExecution(new StartWorkflowExecutionRequest()

{

Input = "{\"inputparam1\":\"value1\"}", // Serialize input to a string

WorkflowId = workflowID,

Domain = domainName,

WorkflowType = new WorkflowType()

{

Name = name,

Version = "2.0"

}

});

Console.WriteLine("Setup: Workflow Instance created ID=" + workflowID);

}

# Worker

static void Worker(string tasklistName)

{

string prefix = string.Format("Worker{0}:{1:x} ", tasklistName,

System.Threading.Thread.CurrentThread.ManagedThreadId);

while (true)

{

Console.WriteLine(prefix + ": Polling for activity task ...");

PollForActivityTaskRequest pollForActivityTaskRequest =

new PollForActivityTaskRequest() {

Domain = domainName,

TaskList = new TaskList()

{

// Poll only the tasks assigned to me

Name = tasklistName

}

};

PollForActivityTaskResponse pollForActivityTaskResponse =

swfClient.PollForActivityTask(pollForActivityTaskRequest);

RespondActivityTaskCompletedRequest respondActivityTaskCompletedRequest =

new RespondActivityTaskCompletedRequest() {

Result = "{\"activityResult1\":\"Result Value1\"}",

TaskToken = pollForActivityTaskResponse.ActivityTask.TaskToken

};

if (pollForActivityTaskResponse.ActivityTask.ActivityId == null)

{

Console.WriteLine(prefix + ": NULL");

}

else

{

RespondActivityTaskCompletedResponse respondActivityTaskCompletedResponse =

swfClient.RespondActivityTaskCompleted(respondActivityTaskCompletedRequest);

Console.WriteLine(prefix + ": Activity task completed. ActivityId - " +

pollForActivityTaskResponse.ActivityTask.ActivityId);

}

}

}

# Decider

static void ScheduleActivity(string name, List<Decision> decisions)

{

Decision decision = new Decision()

{

DecisionType = DecisionType.ScheduleActivityTask,

ScheduleActivityTaskDecisionAttributes = // Uses DefaultTaskList

new ScheduleActivityTaskDecisionAttributes() {

ActivityType = new ActivityType()

{

Name = name,

Version = "2.0"

},

ActivityId = name + "-" + System.Guid.NewGuid().ToString(),

Input = "{\"activityInput1\":\"value1\"}"

}

};

Console.WriteLine("Decider: ActivityId=" +

decision.ScheduleActivityTaskDecisionAttributes.ActivityId);

decisions.Add(decision);

}

// Simple logic

// Creates four activities at the begining

// Waits for them to complete and completes the workflow

static void Decider()

{

int activityCount = 0; // This refers to total number of activities per workflow

IAmazonSimpleWorkflow swfClient = AWSClientFactory.CreateAmazonSimpleWorkflowClient();

while (true)

{

Console.WriteLine("Decider: Polling for decision task ...");

PollForDecisionTaskRequest request = new PollForDecisionTaskRequest()

{

Domain = domainName,

TaskList = new TaskList() {Name = "HelloWorld"}

};

PollForDecisionTaskResponse response = swfClient.PollForDecisionTask(request);

if (response.DecisionTask.TaskToken == null)

{

Console.WriteLine("Decider: NULL");

continue;

}

int completedActivityTaskCount = 0, totalActivityTaskCount = 0;

foreach (HistoryEvent e in response.DecisionTask.Events)

{

Console.WriteLine("Decider: EventType - " + e.EventType +

", EventId - " + e.EventId);

if (e.EventType == "ActivityTaskCompleted")

completedActivityTaskCount++;

if (e.EventType.Value.StartsWith("Activity"))

totalActivityTaskCount++;

}

Console.WriteLine(".... completedCount=" + completedActivityTaskCount);

List<Decision> decisions = new List<Decision>();

if (totalActivityTaskCount == 0) // Create this only at the begining

{

ScheduleActivity("Activity1A", decisions);

ScheduleActivity("Activity1B", decisions);

ScheduleActivity("Activity2", decisions);

ScheduleActivity("Activity2", decisions);

activityCount = 4;

}

else if (completedActivityTaskCount == activityCount)

{

Decision decision = new Decision()

{

DecisionType = DecisionType.CompleteWorkflowExecution,

CompleteWorkflowExecutionDecisionAttributes =

new CompleteWorkflowExecutionDecisionAttributes {

Result = "{\"Result\":\"WF Complete!\"}"

}

};

decisions.Add(decision);

Console.WriteLine("Decider: WORKFLOW COMPLETE!!!!!!!!!!!!!!!!!!!!!!");

}

RespondDecisionTaskCompletedRequest respondDecisionTaskCompletedRequest =

new RespondDecisionTaskCompletedRequest() {

Decisions = decisions,

TaskToken = response.DecisionTask.TaskToken

};

swfClient.RespondDecisionTaskCompleted(respondDecisionTaskCompletedRequest);

}

}

# Output

Setup: Created Domain - HelloWorldDomain

Setup: Created Activity Name - Activity1A

Setup: Created Activity Name - Activity1B

Setup: Created Activity Name - Activity2

Setup: Registerd Workflow Name - HelloWorld Workflow

WorkerActivity1:b : Polling for activity task ...

WorkerActivity1:c : Polling for activity task ...

WorkerActivity2:10 : Polling for activity task ...

WorkerActivity2:11 : Polling for activity task ...

Decider: Polling for decision task ...

Setup: Workflow Instance created ID=Hello World WorkflowID - 635372185191919308

Decider: EventType - WorkflowExecutionStarted, EventId - 1

Decider: EventType - DecisionTaskScheduled, EventId - 2

Decider: EventType - DecisionTaskStarted, EventId - 3

.... completedCount=0

Decider: ActivityId=Activity1A-175b5973-2915-40af-8da3-be970797d401

Decider: ActivityId=Activity1B-fe3eea16-e45e-42d0-b43c-d2bb105ed0c6

Decider: ActivityId=Activity2-77fca339-09b3-4cd8-ad64-86274de50d89

Decider: ActivityId=Activity2-71d972be-a835-41d0-afac-a1d722faa85d

Decider: Polling for decision task ...

WorkerActivity2:10 : Activity task completed. ActivityId - Activity2-77fca339-09b3-4cd8-ad64-86274de50d89

WorkerActivity2:10 : Polling for activity task ...

WorkerActivity1:c : Activity task completed. ActivityId - Activity1A-175b5973-2915-40af-8da3-be970797d401

WorkerActivity1:c : Polling for activity task ...

WorkerActivity1:b : Activity task completed. ActivityId - Activity1B-fe3eea16-e45e-42d0-b43c-d2bb105ed0c6

WorkerActivity1:b : Polling for activity task ...

WorkerActivity2:11 : Activity task completed. ActivityId - Activity2-71d972be-a835-41d0-afac-a1d722faa85d

WorkerActivity2:11 : Polling for activity task ...

Decider: EventType - WorkflowExecutionStarted, EventId - 1

Decider: EventType - DecisionTaskScheduled, EventId - 2

Decider: EventType - DecisionTaskStarted, EventId - 3

Decider: EventType - DecisionTaskCompleted, EventId - 4

Decider: EventType - ActivityTaskScheduled, EventId - 5

Decider: EventType - ActivityTaskScheduled, EventId - 6

Decider: EventType - ActivityTaskScheduled, EventId - 7

Decider: EventType - ActivityTaskScheduled, EventId - 8

Decider: EventType - ActivityTaskStarted, EventId - 9

Decider: EventType - ActivityTaskStarted, EventId - 10

Decider: EventType - ActivityTaskStarted, EventId - 11

Decider: EventType - ActivityTaskStarted, EventId - 12

Decider: EventType - ActivityTaskCompleted, EventId - 13

Decider: EventType - DecisionTaskScheduled, EventId - 14

Decider: EventType - ActivityTaskCompleted, EventId - 15

Decider: EventType - ActivityTaskCompleted, EventId - 16

Decider: EventType - ActivityTaskCompleted, EventId - 17

Decider: EventType - DecisionTaskStarted, EventId - 18

.... completedCount=4

Decider: WORKFLOW COMPLETE!!!!!!!!!!!!!!!!!!!!!!

Decider: Polling for decision task ...

# Code

The code along with Visual Studio solution can be found under “AWS” folder at <https://github.com/padisetty/Samples>.

Explore & Enjoy!

/Siva