Crossplane

CONTRIBFEST · KUBECON + CLOUDNATIVECON NORTH AMERICA 2026

Start your journey to Crossplane function maintainer

From an empty repo to a PR-ready function in 80 minutes

Yury Tsarev
Principal Solutions Architect, Upbound

Salt Palace · Level 2 · 255 D
Nov 11 · 11:00 am–12:20 pm

BEFORE WE START

Check your setup

Go 1.26+, then run the setup script. No Docker needed.
You're set when it says You're ready for the labs.

```
git clone https://github.com/upbound/crossplane-contribfest-slc
cd crossplane-contribfest-slc
./scripts/setup.sh
```



80 minutes, mostly at the keyboard

Time	What we do
5 min	Check your setup
10 min	Why functions, the ecosystem, and today’s problem
41 min	Labs 1–4: scaffold, core logic, input API, debugging
10 min	Lab 5: run it as an Operation, then make it PR-ready
10 min	Lab 6: test it on a local cluster, the last check before a PR
4 min	From first PR to maintainer, and what to pick up next

Fell behind? Every lab ends at a git tag. One command catches you up.

Functions are where your platform logic lives

</>

Any logic

Loops, conditionals, and real code in Go, Python, TypeScript, KCL, or templates.

Live data

Query cloud APIs and other resources, then compose from what you find.

Policy

Rewrite what earlier steps composed: tags, defaults, lifecycle.

In Crossplane v2, Compositions and Operations both run a pipeline of functions.

The pipeline is middleware

STEP 1

compose

`function-patch-and-transform`

Adds a Bucket, an Object, and a ConfigMap

STEP 2 · TODAY

lifecycle

`function-lifecycle`

Rewrites managementPolicies on the managed resources

STEP 3

auto-ready

`function-auto-ready`

Marks the XR ready when its resources are

⚠ Each step gets the desired state so far and returns it. Leave a resource out, and Crossplane deletes it.

A tour of the ecosystem

Every name below is a function you can install today: function-…

Write it your way

- go-templating
- kcl
- python
- cue

SDKs: Go, Python, TypeScript, Java, Rust

Glue the pipeline

- auto-ready
- environment-configs
- extra-resources
- sequencer
- status-transformer
- cel-filter
- tag-manager
- deletion-protection

Solve one problem

- aws-query
- azresourcegraph
- msgraph
- … and the one you write today

Some functions query live cloud state

`function-aws-query`

Read-only AWS: regions, AMIs, quotas, Cloud Control, subnets

`function-azresourcegraph`

KQL queries over Azure Resource Graph

`function-msgraph`

Users, groups, and service principals from Microsoft Graph

```
queryRef: context.[apiextensions.crossplane.io/environment].query
target:   status.azResourceGraphQueryResult
```

Read from spec., status., or context. and write to status. or context. You'll reuse this grammar in Lab 3.

A function is one gRPC call

```
rpc RunFunction(RunFunctionRequest) returns (RunFunctionResponse)
```

Request

- observed: what exists now
- desired: what earlier steps want
- input: this step's config
- context: data from earlier steps
- required resources and schemas
- credentials

Response

- desired: what you want, changed or not
- results: events on the XR
- conditions: status on the XR
- context: data for later steps
- requirements: ask for more resources
- TTL: when to call you again

The same call serves Composition and Operation functions. Your package declares which.

Management policies decide what Crossplane may do

`Observe` `Create` `Update` `Delete` `LateInitialize` `* = all of them`

15 sets

crossplane-runtime supports 15 combinations. Anything else is rejected.

No deletionPolicy

v2 namespaced MRs dropped it. Policies are now the only way to orphan.

Beta, on by default

In upjet providers and provider-kubernetes. Off in classic provider-aws.

We hand-rolled this five times

```
managementMode: Full | Provision | ObserveOnly | Deprovision
["Observe"] if observe_only else ["*"]
management_policies(deletion_policy)
["Observe", "Create", "Update"]  # never delete
params.managementPolicies or ["*"]
```

Five copies, five spellings, and no single switch an operator can flip.

The community has been discussing this

crossplane#6694	An Orphan management policy	Added, then reverted
crossplane#7146	Orphan composed resources from a Composition	Closed as stale
crossplane#7367	managementPoliciesOptions: an XR-level orphan switch	Open draft

Policy across a composition is a pipeline concern. That's what functions are for.

function-lifecycle: one switch per XR

Mode	managementPolicies
Full	["*"]
Provision	[Observe, Create, Update, LateInitialize]
ObserveOnly	[Observe]
Deprovision	[Observe, Create, Delete]

Where the mode comes from

1. The XR annotation
2. modeRef: spec., status., or context.
3. The input's mode
4. Full

An unknown mode is a fatal error. It never falls back to Full.

One annotation in, policies out

THE XR

```
apiVersion: example.crossplane.io/v1
kind: App
metadata:
  name: my-app
  annotations:
    lifecycle.fn.crossplane.io/mode: Provision
```

THE RENDERED BUCKET

```
kind: Bucket
spec:
  forProvider:
    region: us-west-2
  managementPolicies:
  - Observe
  - Create
  - Update
  - LateInitialize
```

The Bucket and the provider-kubernetes Object get the policies. The plain ConfigMap is skipped: it has no spec.forProvider.

LAB 1 OF 6

8 min

Scaffold and render

```
cd ~ && crossplane xpkg init function-lifecycle \
  function-template-go -d function-lifecycle -r
cd function-lifecycle
# renames and the 127.0.0.1 target: see the page
git init -b main && git add -A
git commit -s -m "chore: scaffold"
# terminal 1
go run . --insecure --debug --address=127.0.0.1:9443
# terminal 2, in ~/function-lifecycle too
crossplane composition render example/xr.yaml \
  example/composition.yaml example/functions.yaml \
  -r --crossplane-binary=$(which crossplane-core)
```

Done when

Render prints I was run with input “Hello world”!

Stuck?

Raise your hand, or start over from workshop/lab-01 with the lab page’s catch-up

Lab page: github.com/upbound/crossplane-contribfest-slc/blob/main/labs/01-scaffold.md

LAB 2 OF 6

15 min

Core logic, tests first

```
cd ~/function-lifecycle
REF=https://github.com/upbound/function-lifecycle
git fetch $REF workshop/lab-02-start
git checkout FETCH_HEAD -- fn.go mode.go fn_test.go example/
# terminal 3, leave it running
function-patch-and-transform --insecure \
  --address=127.0.0.1:9444
# fn.go: fill in the three TODO(lab-02)
go test -v ./...
# restart terminal 1, render in terminal 2
crossplane composition render example/xr-provision.yaml \
  example/composition.yaml example/functions.yaml \
  -r --crossplane-binary=$(which crossplane-core)
```

Done when

Tests pass, and render says 2 managed resources (1 skipped)

Stuck?

Jump to workshop/lab-02 with the lab page's catch-up

Lab page: github.com/upbound/crossplane-contribfest-slc/blob/main/labs/02-core-logic.md

LAB 3 OF 6

10 min

Generalize with an input API

```
cd ~/function-lifecycle
REF=https://github.com/upbound/function-lifecycle
git fetch $REF workshop/lab-03-start
git checkout FETCH_HEAD -- fn.go fn_test.go input/ example/
go generate ./...
# fn.go: fill in the four TODO(lab-03)
go mod tidy && go test -v ./...
# restart terminal 1, render in terminal 2
crossplane composition render example/xr-spec-mode.yaml \
  example/composition.yaml example/functions.yaml \
  -r --crossplane-binary=$(which crossplane-core)
```

Done when

The spec render shows ObserveOnly, and the page's context render shows Deprovision

Stuck?

Jump to workshop/lab-03 with the lab page's catch-up

Lab page: github.com/upbound/crossplane-contribfest-slc/blob/main/labs/03-generalize-input.md

LAB 4 OF 6

8 min

Make it explain itself

```
cd ~/function-lifecycle
REF=https://github.com/upbound/function-lifecycle
git fetch $REF workshop/lab-04-start
git checkout FETCH_HEAD -- fn.go fn_test.go example/
# fn.go TODO(lab-04) 1: where the mode came from
# fn.go TODO(lab-04) 2: the LifecycleMode condition
go test -v ./...
# restart terminal 1, then in terminal 2
./example/test.sh
```

Done when

example/test.sh prints PASS for all seven examples

Stuck?

Jump to workshop/lab-04 with the lab page's catch-up

Lab page: github.com/upbound/crossplane-contribfest-slc/blob/main/labs/04-pipeline-debug.md

LAB 5 OF 6

10 min

Run it as an Operation, then PR-ready

```
cd ~/function-lifecycle
REF=https://github.com/upbound/function-lifecycle
git fetch $REF workshop/lab-05-start
git checkout FETCH_HEAD -- fn.go fn_test.go mode.go \
  operation.go operation_test.go example/ README.md \
  package/crossplane.yaml
# fill in the three TODO(lab-05), restart terminal 1
crossplane operation render example/operation.yaml \
  example/functions.yaml -e example/apps.yaml \
  -r --crossplane-binary=$(which crossplane-core)
go fix ./...   # then delete ptr: 0 lint issues
git switch -c pr-ready && git add -A
git commit -s -m "feat: add lifecycle modes per XR"
```

Done when

The Operation patches 2 of 3 Apps, and pr-ready has a signed-off commit

Done early?

Take a contribfest issue on upbound/function-lifecycle

Lab page: github.com/upbound/crossplane-contribfest-slc/blob/main/labs/05-pr-ready.md

LAB 6 OF 6

10 min

Local e2e, the last check before a PR

```
cd ~/function-lifecycle
REF=https://github.com/upbound/function-lifecycle
git fetch $REF workshop/lab-06
git checkout FETCH_HEAD -- e2e/
# check the control plane, then install your
# function by hand: image, package, Function
# try the modes, then the CronOperation by hand
./e2e/run.sh   # the same install, then: PASS
# delete "- operation" in package/crossplane.yaml
./e2e/run.sh   # FAIL: missing required capabilities
# put it back, run again: PASS
git add e2e/
git commit -s -m "test: add a local e2e test on kind"
```

Done when

The e2e test passes and is committed: your branch is PR-ready

No Docker?

Pair up with a neighbor, or watch the presenter run it on the projector

Lab page: github.com/upbound/crossplane-contribfest-slc/blob/main/labs/06-e2e-stretch.md

What a maintainer checks first

The code

- ✓ A test for every behavior
- ✓ golangci-lint: 0 issues
- ✓ go generate and go mod tidy: no diff
- ✓ Every example renders
- ✓ The package builds

The people

- ✓ A README a user can follow
- ✓ Package metadata filled in
- ✓ Every commit signed off (-s)
- ✓ The PR template, checklist and all
- ✓ “Fixes #N” links the issue

Using an AI assistant? Read AI_POLICY.md

Assumed

No disclosure needed. Leave Co-authored-by trailers for agents out.

Own every line

If you can't explain a change without the agent, it isn't ready.

Talk to us

Your reasoning, in your words, at the right level of detail.

github.com/crossplane/crossplane/blob/main/AI_POLICY.md · applies to crossplane and crossplane-contrib

See inside the pipeline

Tool	What it shows
render -r	Results from every step, in order
render -c	The pipeline context after the last step
render -x	The input XR's spec and metadata, in the output
render -o observed.yaml	What happens when resources already exist
go run . --debug	Your f.log.Debug lines
crossplane resource trace	The live XR tree and its conditions
Pipeline inspector (alpha)	Every request and response, in a cluster

Same contract, any language

Go

`function-sdk-go v0.7`

- Template and docs guide
- xpkg init knows it
- Today's labs

Python

`function-sdk-python v0.15`

- Template and docs guide
- xpkg init knows it
- Bigger image, same contract

Rust

`function-sdk-rust v0.4`

- Official since August 2026
- No template yet
- Your contribution?

Also official: function-sdk-typescript and function-sdk-java. One gRPC contract, so every lab today maps one to one.

From first PR to maintainer

Use
File issues, answer questions

→

Contribute
PRs, docs, triage. You are here.

→

Review
Review others' PRs, own an area

→

Maintain
Listed in OWNERS.md, approve and merge

Each repo's maintainers choose new maintainers. They look for consistent leadership, domain expertise, solid code and tests, and care for the whole repo.

Crossplane GOVERNANCE.md · Becoming a Maintainer

Pick your next contribution

function-lifecycle

- A raw policies escape hatch
- respectExisting
- Propagate to nested XRs
- Debug output in the context
- Most restrictive mode wins
- Expiry and change windows

Across the ecosystem

- cli: render's 10 s localhost delay
- cli: xpkg init through a symlink
- template-go #74: init.sh renames
- template-go: deprecated gomodguard
- template-go #82: already done, close it
- go-templating #61, #40
- environment-configs #6
- template-typescript: SDK 0.8

All of them: github.com/upbound/crossplane-contribfest-slc/blob/main/CURATED-ISSUES.md

Your next function

Stable secrets

Generate a password once and keep it stable by reading it back from observed state. Nothing on the Marketplace does this yet.

Expiry checks

A CronOperation that finds certificates about to expire and flags them. You built an Operation step today, so you know the shape.

A Rust template

The official Rust SDK has no function-template-rust yet. Build the one everyone after you starts from.

Started from a problem you copy-pasted? That’s a function.

Thank you

Keep going. Every lab, issue, and link is in the hub repo.

`github.com/upbound/crossplane-contribfest-slc`

`github.com/upbound/function-lifecycle`

`slack.crossplane.io`



Rate this session