Crossplane

Contribfest

github.com/crossplane-contrib/contribfest

Nic Cope, Upbound
Jared Watts, Upbound

Agenda & Goals

- Grow and empower a high quality contributor base
    - All projects can benefit from more great contributors
- Enhance understanding of Composition Functions
    - What do they solve? How do they work? What can they do?
- How to write your own Composition Function
    - Accelerate your understanding and adoption of this powerful new Crossplane feature
- Get direct feedback from you
    - How can we improve the developer experience in Crossplane?

Composition Functions Overview

Crossplane

Current Limitations of Composition

- No iteration. No conditionals. No templates.
- No advanced logic other than simple patch & transforms
- List of resources is static
- No ability to call external APIs to get values
- and others...composition is **not** a programming language
- We didn’t want to grow a DSL expressed in YAML
  - Need to reinvent a lot of wheels - testing, linting, etc
  - Infra DSLs tend to grow organically, but not cohesively
  - We’re not language designers

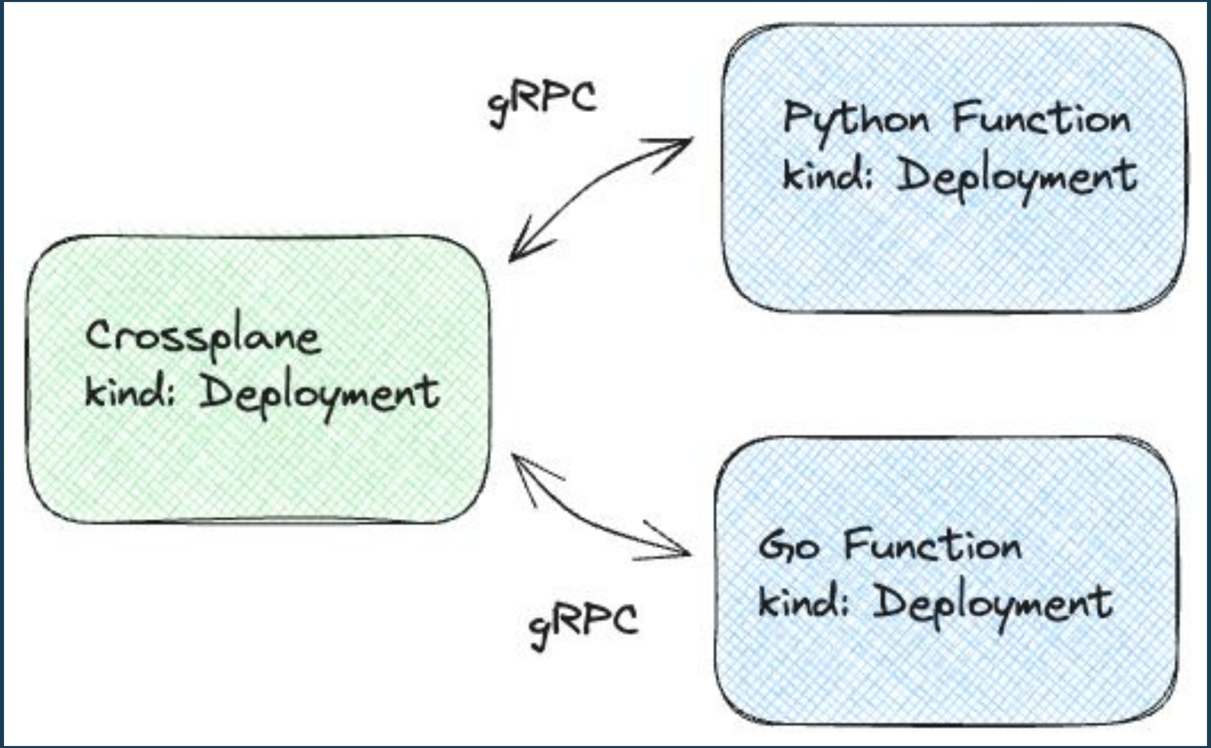
What can Functions do?

- First released as **alpha** in **v1.11.0**
- Evolved the architecture and UX to mature to **beta** in **v1.14.0**
- Run a pipeline of simple functions
- Written in your language of choice with any logic your use case needs
- You don't **have** to write any code to start using functions
    - Reusable functions that are generally useful
    - e.g., helm/go templates, CUE scripts, etc.
- Sweet spot between "no code" ←→ building an entire controller
    - Focus on your platform's unique needs only
    - Crossplane still does the heavy lifting of CRUD-ing resources, finalizers, owner refs, etc

How do Functions work?

- Packaged/distributed just like `Providers` and `Configurations`
- Each function is more like a “function server”
  - Long running processes, created as a `Deployment` and headless `Service`
- Crossplane talks gRPC to each function
  - `RunFunctionRequest`, `RunFunctionResponse`
  - secured by mutual transport layer security (mTLS)
- Pipeline of data passing from one function to the next, each modifying and validating the stream as needed
- Functions don’t interact with API server - Crossplane will take resulting set of resources and “make it so”

Functions Visualized



Building Functions

- SDKs
  - Libraries for common tasks, codify best practices, eliminate boilerplate
  - e.g., https://github.com/crossplane/function-sdk-go
- Tooling
  - Scaffold a new Function with everything except your custom logic
    - `crossplane beta xpkg init myfunc function-template-go`
  - Test and iterate locally
    - `crossplane beta render xr.yaml composition.yaml functions.yaml`
  - Build/Push to package registry
    - `crossplane xpkg build`
    - `crossplane xpkg push myorg/cool-func:v0.1.0`

Template project for `go`

- Everything you need to get started writing a Function in `go`
    - https://github.com/crossplane/function-template-go
- `main.go` entry point sets up long running function server
- `Dockerfile` to build the function runtime
- `package` dir defines Crossplane `Function` package/metadata
- API for input/config to your function, e.g., `inputs.template.fn.crossplane.io/v1beta1`
- `fn.go` - write your code in `RunFunction()`
- `fn_test.go` unit testing for your specific logic

Crossplane

SDK for go

- Useful helper packages to streamline function development
  - https://github.com/crossplane/function-sdk-go
- `request` - utilities for working with `RunFunctionRequests`
- `response` - utilities for working with `RunFunctionResponses`
- `resource` - work with Crossplane resources, getters/setters and conversions for golang ←→ protobuf
  - `composite` - work with a composite resource (XR)
  - `composed` - work with composed resources
- `errors` - handle and report errors
- `logging` - log useful information and events

Crossplane

Hands-on Lab

Writing a Function in go

https://github.com/crossplane-contrib/contribfest

lab-01.md

#contribfest on slack.crossplane.io

Crossplane

Bonus: Hands-on Lab

Writing your own Function

https://github.com/crossplane-contrib/contribfest lab-02.md

Get creative!

Crossplane

Community is everything

Get Involved

- Website: https://crossplane.io/
- Docs: https://docs.crossplane.io/
- GitHub: https://github.com/crossplane/crossplane
- Slack: https://slack.crossplane.io/
- Blog: https://blog.crossplane.io/
- Twitter: https://twitter.com/crossplane_io
- Youtube: Crossplane Youtube

Calling all Crossplane Adopters!

🚨 We'd love to hear about your adoption of Crossplane, please share your story in ADOPTERS.md in the crossplane/crossplane repo 🚨

Crossplane