Get Started with GitOps

Operations by Pull Request enable Terraform for Teams

Cloud Posse
<hello@cloudposse.com>
https://cloudposse.com/
@cloudposse
What to Expect

Feelings of Euphoria
Aha! Moments
Reduced Anxiety

AND...

What is GitOps? (not rocket science)
Why it’s awesome (and you’ll agree)
How to get started... (our way)

+ HashiConf News!

LIVE DEMO

Q&A
Who is this dude?

Founder of **Cloud Posse** a DevOps Professional Services Company

We’ve pioneered **SWEETOPS**

Collaborative DevOps for Companies

**(100% OPEN SOURCE)**

(cloudposse.com)
We got problems.

We Maintain **100+ Terraform Modules** (the largest!)

Dozens of Helm Charts

Pain in the *ss to **test everything**

Multi-stage **rollouts get complicated**

Lots of tools like Helm, Kops, Terraform and Cloud Formation

Thousands of users  (hey, some problems are good to have!)
Goal:

Effortlessly Deploy Infrastructure

(e.g. spin up RDS Database with Terraform, or deploy Helm Chart)
One Approach...

Make changes in the privacy of your personal laptop.
(sometimes after a few beers)
“I ^ it worked on my machine.”
Then comes... **LAUNCH DAY**

It worked and tested in dev

Now it's devops problem

Production
Other Problems...

No Audit Trails (huge risk)

Complicated Manual Rollouts

Not clear what’s been deployed (configuration drift)

Failed Deployments on Merge (NOW WHAT?!)  

Insufficient Code Reviews

No one knows how to make changes
So....
Let's fix this.
Let’s Practice GitOps.

Use **Git as a System of Record** for the desired state of configuration.

Do **Operations by Pull Request** for **Infrastructure as Code**.

Then use **Continuous Delivery to apply changes** to infrastructure (basically it’s a CI/CD for DevOps).

Issue **commands using comments** to trigger actions (a.k.a “ChatOps”).

(E.g. “@bot give me a plan”, “@bot deploy these changes”)

- Run **PLAN**
  - See what should change
- Run **APPLY**
  - See what actually happened
The "Git Workflow"

1. Developer commits changes to a feature branch.
2. Pull Request is opened.
3. Pull Request is reviewed.
4. Changes are requested if necessary.
5. Pull Request is merged to master.
Why do you care?
Teamwork.
GitOps Objectives

**Repeatable** - Apply changes the same way every time (even your entire stack all at once!)

**Predictable** - Know what’s going to happen (e.g. before you merge)

**Auditable** - See what was done (e.g. when things were applied. see if there were errors)

**Accessible** - Anyone who can open a PR can contribute
The Solution

Now an official HashiCorp sponsored project
Built for Terraform
(but will run anything)
About Atlantis

Purpose-built for Terraform (understands init, plan, apply)

Project started at hootsuite

Officially forked into https://github.com/runatlantis/atlantis

Current Maintainer is Luke Kysow

Open Source APACHE2

100% Golang with good test coverage
Basic Flow Diagram

Developer pushes commits to GitHub. GitHub triggers a HTTPS webhook using a secret token. Terraform is set up to automate the process in the cloud. The Atlantis Server/Container interacts with Amazon EC2 and Amazon VPC, adding comments with output.
How We Use Atlantis

Terraform
Cloud Formation
Helm
Helmfile

Because we can run any command

But wait... there's more!
But will it work with...

TERRAGRUNT? YES

GITLAB? YES

BITBUCKET? YES

DOCKER? YES

BUT WAIT! THERE'S MORE!
“Interactive”
Pull Requests
Step One: Open Pull Request
Step Two: Review “Auto Plan”
Step Three: Seek Approval

**CODE REVIEW**

- **Review required**: At least 1 approving review is required by reviewers with write access. Learn more.
- **All checks have passed**: 2 successful checks
  - **Atlantis — Plan Success**
  - **root.cloudposse.co — Build passed**
- **Merging is blocked**: Merging can be performed automatically with 1 approving review.

- **aknysh** approved these changes 35 seconds ago
Step Four: Deploy Changes

osterman commented 5 minutes ago

atlantis/root apply

cloudpossebot commented 2 minutes ago

Ran Apply in dir: conf/users workspace: default

Apply complete! Resources: 3 added, 0 changed, 0 destroyed. Releasing state lock. This may take a few moments...

Outputs:

account_alias = cppo-root-account
aknysh_decrypt_command = echo "wcFMAyI0j/2+yeBZARAAYED+UcMl+QBBnY+l3kvnnGEHS7nOY0fBUyZie7

osterman_decrypt_command = echo "wcFMAyI0j/2+yeBZARAAYED+UcMl+QBBnY+l3kvnnGEHS7nOY0fBUyZie7"
Step Five: Merge Pull Request

Pull request successfully merged and closed
You’re all set—the add-aknysh branch can be safely deleted.

Nailed It!
That was easy.
Atlantis Users?

(...and soon most of our customers) 😊
We prevent #terraform changes to our infrastructure that haven’t been approved by a reviewer with the "require-approval:true" flag on the #atlantis config, this makes our PR process 👍🔒 @runatlantis

natalysheinlin commented just now
atlantis apply -d terraform/clusters/shopify-gke-bugbounty

sa-atlantis commented just now
Apply Failed: Pull request must be approved before running apply.

2:22 AM - 20 Jul 2018
The live demo of Atlantis was extra dope. @hootsuite should be proud to have sent one of their best. [https://runatlantis.io](https://runatlantis.io)
How to get started

1. Deploy Atlantis (e.g. ECS, Kubernetes+Helm)
2. Add `atlantis.yaml` to each repo
3. Get back to work *(sorry)*.

Or just ask us for help =)
Deploy Atlantis on ECS Fargate

- fargate certificate create
- fargate certificate validate
- fargate lb create
- fargate lb alias
- fargate service create

1. Create TLS Certificate
2. Activate it
3. Create Load Balancer
4. Assign DNS
5. Deploy Container

https://github.com/cloudposse/geodesic-aws-atlantis
Example: `atlantis.yaml`

```yaml
version: 2
projects:
  - name: "alpinist"
    dir: "terraform"
    workspace: "default"
    terraform_version: "v0.11.7"
  autoplan:
    when_modified:
      - "*.tf"
    enabled: true
  apply_requirements:
    - "approved"
  workflow: "default"

# define list of chart repositories
# list of projects in this repo
# friendly name for this project
# directory with the tf code
# workspace to use with this project
# terraform version to use
# automatically run `terraform plan`
# when...
# any .tf file changes
# and enabled
# then run `terraform apply`
# only when `approved`
# run this workflow

Steps can be entirely customized.
```
Example atlantis.yaml (Continued)

```
workflows:
  default:
    plan:
      steps:
      - run: "init-terraform"
      - run: |
          terraform plan -no-color \n          -var-file atlantis.tfvars -out $PLANFILE
    apply:
      steps:
      - run: |
          terraform apply -no-color \n          -var-file atlantis.tfvars $PLANFILE
```

# define list of workflows
# friendly name for this workflow
# to do a plan
#   perform some steps
#     run a command to initialize tf state
# use fancy YAML conventions
# run a terraform plan use with -var-file
# save the plan to a file for later
# to do a plan...
# run these steps..
# with some fancy YAML
# run apply using previous plan
# $PLANFILE ensures WYSIWYG

Steps can be entirely customized.
1. **ADD USER**
2. **OPEN PR**
3. **RUN PLAN**
4. **SEEK APPROVAL (OR NOT)**
5. **APPLY**
6. **MERGE**
Demo Time!

DEMO GODS

PLEASE LET THIS DEMO WORK
Our Best Practices

Use **one Atlantis Server per account** (prod, dev, staging, identity, security, etc)

Use **IAM Service Account** for credentials (not hardcoded credentials)

Use GitHub **CODEOWNERS**

Use **-var-files** for non-secrets

Use **chamber** by segmentio for secrets (SSM+KMS)

Disable for forks
Gotchas

Atlantis is under active development

We’ve forked it to support **what we needed**

1. Restricted Users
2. Git Submodules
3. Multiple Pipelines (e.g. `atlantis/prod.yaml`, `atlantis/staging.yaml`)
4. Destroy action
5. Custom wake words (e.g. “echo, shut up”)

[https://github.com/cloudposse/atlantis](https://github.com/cloudposse/atlantis)
GitOps

Stop living dangerously.

Start using GitOps.

- Practice total transparency in operations
- Increase **Productivity**, Simplify **Maintenance**, Ensure **Repeatability**
- Reduce the barrier to entry
- Scalable strategy to manage lots of infrastructure

https://github.com/runatlantis/atlantis
HashiConf 2018 Announcements

HashiCorp

Terraform 0.12 (alpha 1) released
+ "Terraform State as a Service"

Vault: Automatic Unsealing -> Open Source

New provider! Manage charts with terraform
Links

Our Fork
https://github.com/cloudposse/atlantis

Our Slack Community
Join our community!
https://slack.cloudposse.com/

Our Demo
https://github.com/cloudposse/root.cloudposse.co
Totally Bodacious

**Geodesic** (container+env pattern for Infrastructure as Code)

github.com/cloudposse/geodesic

**Packages** (our complete toolchain + alpine packages)

github.com/cloudposse/packages

**Build Harness** (Makefiles on Steroids; build anything)

github.com/cloudposse/build-harness

**Reference Architectures**

github.com/cloudposse?q=cloudposse.co

**Documentation**

docs.cloudposse.com
Hire us. =)  

100+ Free Terraform Modules  
Active Community  
Awesome Documentation  

github.com/cloudposse/  
slack.cloudposse.com  
docs.cloudposse.com

415 535 8615  
hello@cloudposse.com  
(free consultation)