Terraform GitOps

How to do Operations by Pull Request

codefresh + HashiCorp Terraform

Cloud Posse

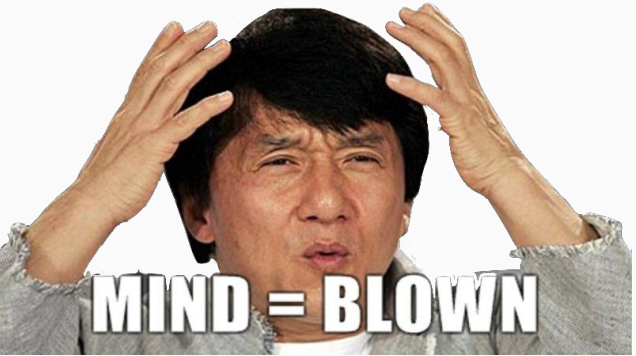
<hello@cloudposse.com>
https://cloudposse.com/
@cloudposse

What to Expect

Feelings of OMG

Aha! Moments...

Totally Sweet Ops



AND...

What is GitOps? (not rocket science)

Why it's awesome (and you'll agree)

How to get started... (our way)

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)

Awesome.

Infrastructure as a Service

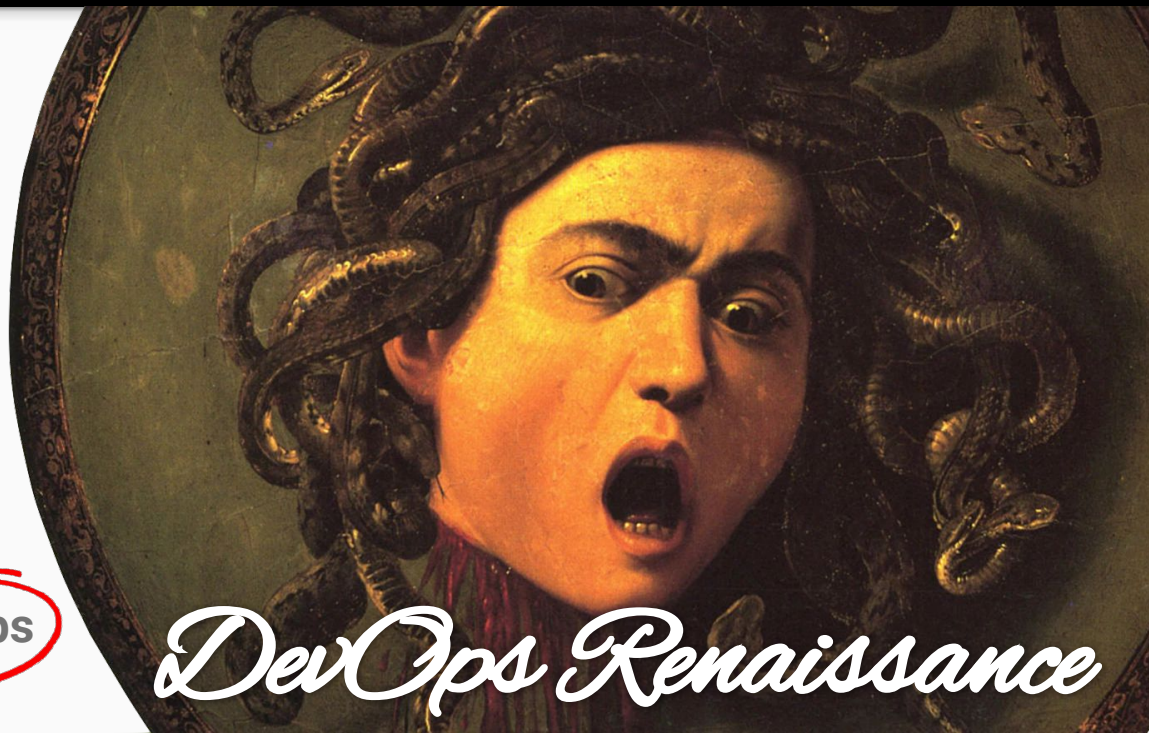
Everything as Code, SDNs

Serverless & Lambdas

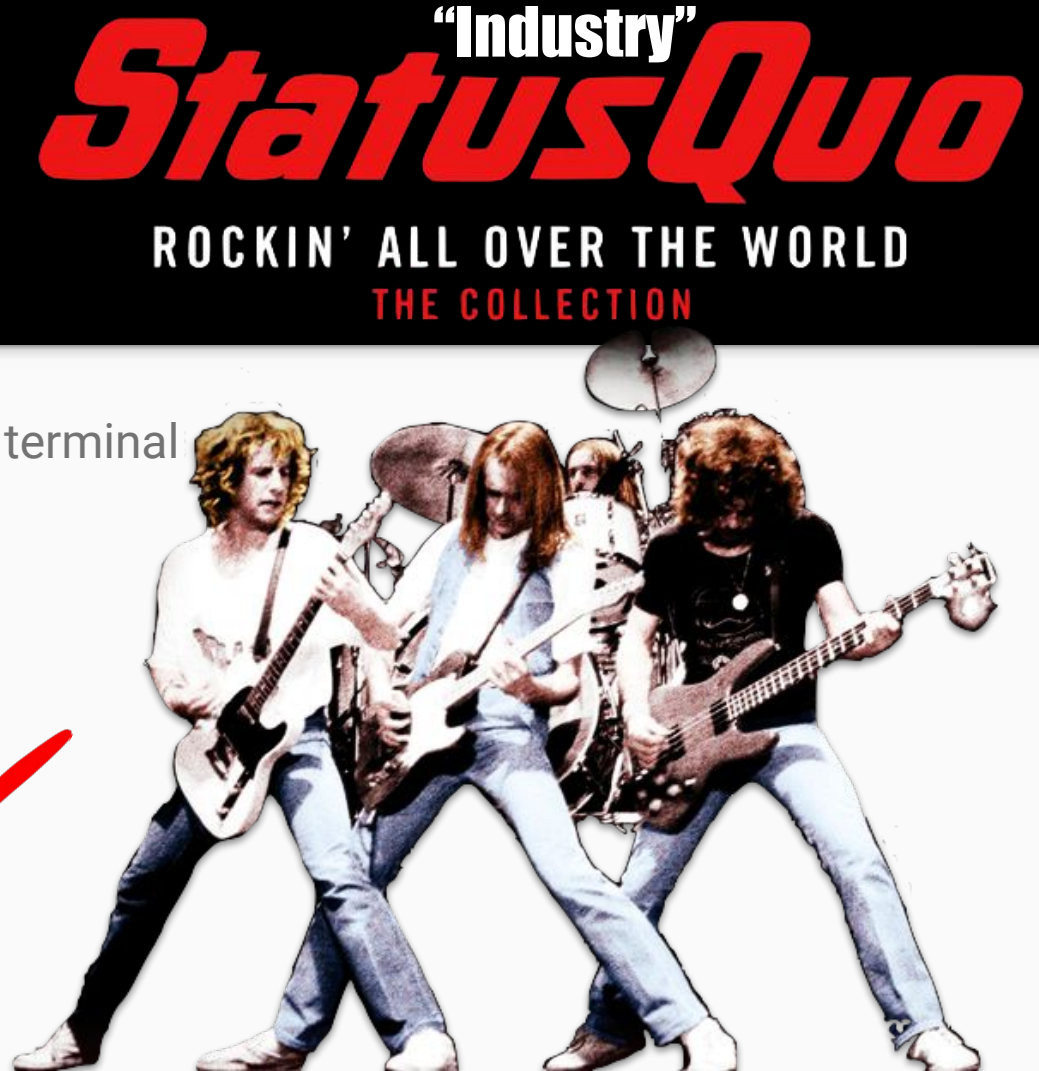
Mesh Networking, Operators

Container Management Platforms
(kubernetes, ecs, mesos, swarm)

CI/CD Everywhere, ChatOps, GitOps



The DevOps



Complicated **Manual Rollouts** via the terminal

Poor Audit Trails (huge risk)

Not clear what's been deployed
(configuration drift)

Out of date documentation

No one knows how to make changes

Terraform more problems

Deploying infrastructure is **not like deploying a web app**
(no easy rollbacks)

Terraform is more like a database migration tool

Terraform **does not automatically rollback** on errors

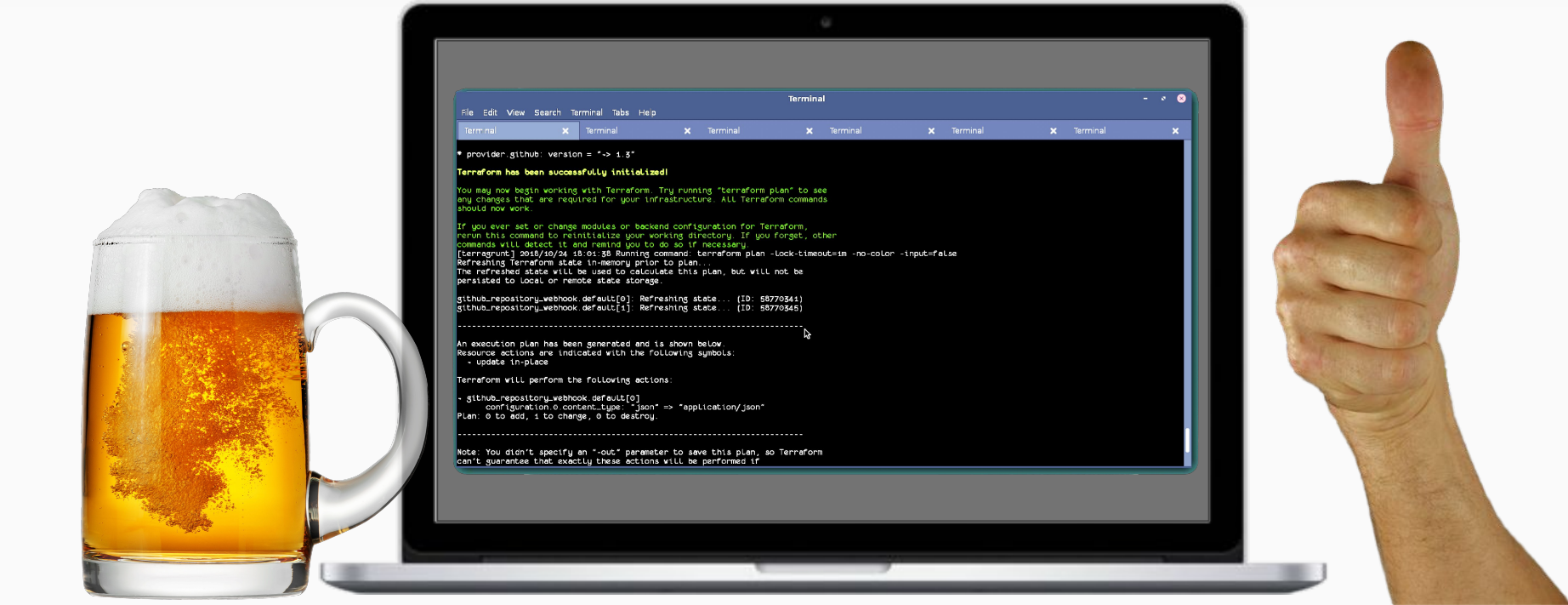
Terraform plans are a **best guess** of what's to happen

Terraform apply will **regularly fail**

Terraform **apply on merge risks destabilizing master**

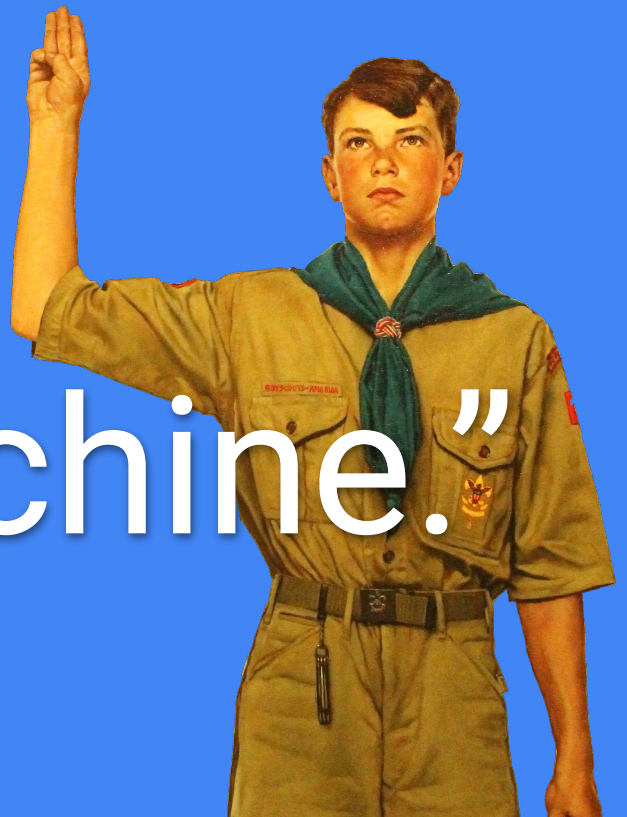
For Example...

I test some changes at home...



SWEAR

“I ^ it worked...
on my machine.”



Then comes... LAUNCH DAY

```
aws_instance.salt_master_a (remote-exec): consul start/running, process 3431
aws_instance.salt_master_a: Creation complete
aws_eip.salt_master_a: Creating...
  allocation_id:  "" => "<computed>"
  association_id: "" => "<computed>"
  domain:         "" => "<computed>"
  instance:       "" => "i-2a1a9afc"
  private_ip:     "" => "<computed>"
  public_ip:      "" => "<computed>"
  vpc:            "" => "1"
aws_eip.salt_master_a: Error: 1 error(s) occurred:

* Failure associating EIP: InvalidAllocationID.NotFound: The allocation ID 'eipalloc-9b0b7cfe' does not exist
aws_route53_record.dns_b: Creation complete
Error applying plan:

1 error(s) occurred:

* 1 error(s) occurred:

* 1 error(s) occurred:

* Failure associating EIP: InvalidAllocationID.NotFound: The allocation ID 'eipalloc-9b0b7cfe' does not exist

Terraform does not automatically rollback in the face of errors.
Instead, your Terraform state file has been partially updated with
any resources that successfully completed. Please address the error
above and apply again to incrementally change your infrastructure.
```

PRODUCTION

The Math is Simple

A*B*C*D*E*F = impossible to manage

A = # of tools pinned to versions
B = # of dependencies pinned to versions
C = # of AWS accounts
D = # of project environments (per acct)
E = # of number of developers
F = # of customers (our case)

TOO MANY PERMUTATIONS TO KEEP STRAIGHT

This is why we don't run things "natively"

THE FINAL FRONTIER

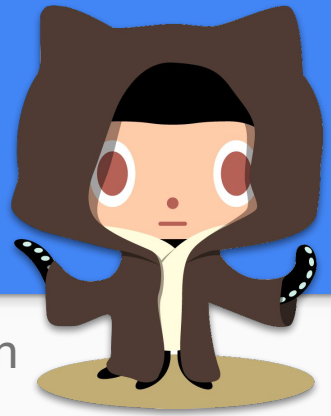
So….
Let’s fix this.

Goal:

Make it Easy to Terraform Stuff

(e.g. enable anyone on team to easily spin up RDS Database with Terraform)

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)

See **output from terraform** in GitHub comments

(E.g. "`Plan: 23 to add, 2 to change, 15 to destroy.`")

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

codefresh

https://codefresh.io

Automate Anything

(**if it runs in a container**)

Because we can
run any command

Terraform

Cloud Formation

Helm → K8S

Helmfile

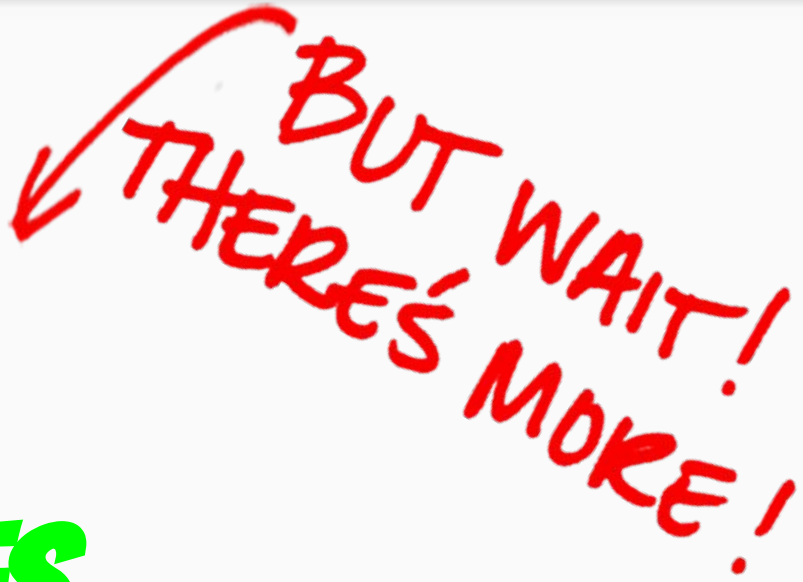
But will it work with...

TERRAGRUNT? YES

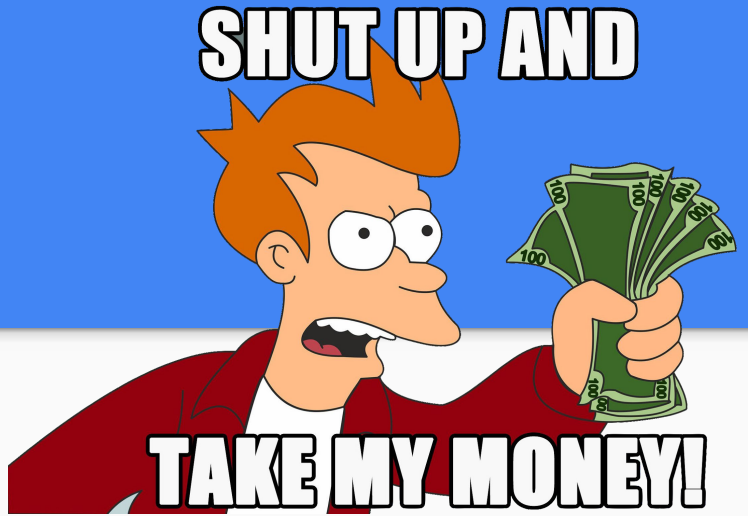
GITLAB? YES

BITBUCKET? YES

ANSIBLE? YES



About Codefresh



Yet another CI/CD solution, only better.

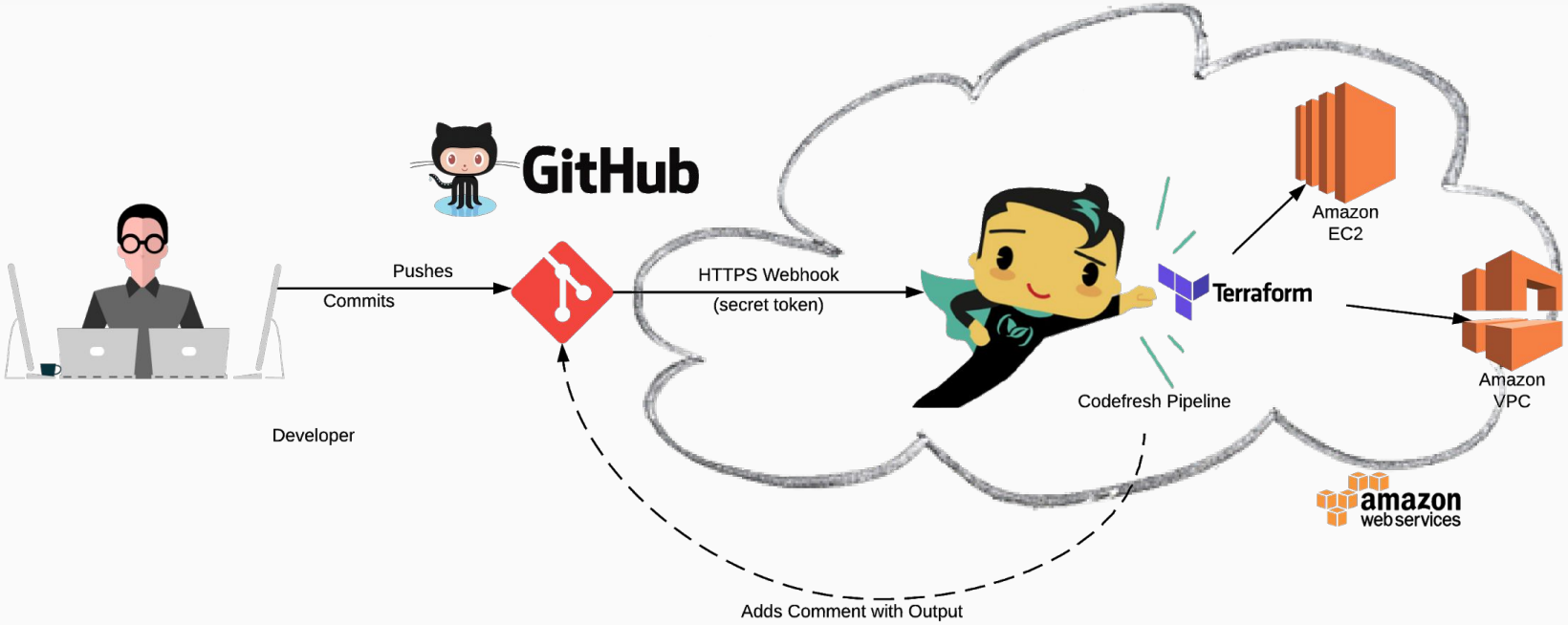
1. **Stick everything you want to automate into containers**
2. **String containers together in a pipeline, run them in parallel**
3. **Trigger pipelines on webhooks, comments, releases, etc.**

Slack Notifications

Approval Steps

GitHub Comments

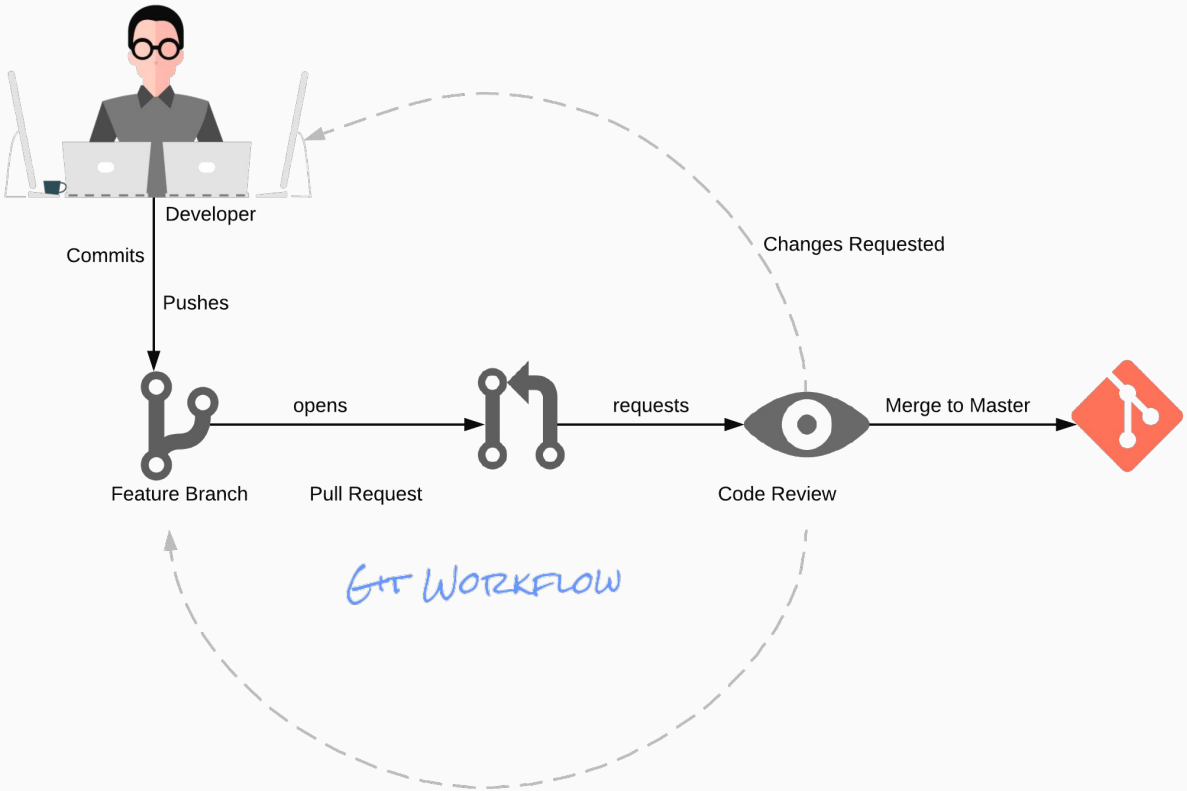
Basic Flow Diagram



“Interactive”

Pull Requests

The “Git Workflow”



Step One: Open Pull Request

Add S3 Bucket #74

Open **osterman** wants to merge 1 commit into `master` from `example-1`

Conversation 1 · Commits 1 · Checks 0 · Files changed 1

osterman commented 2 minutes ago • edited — Member

what

- Add an S3 bucket

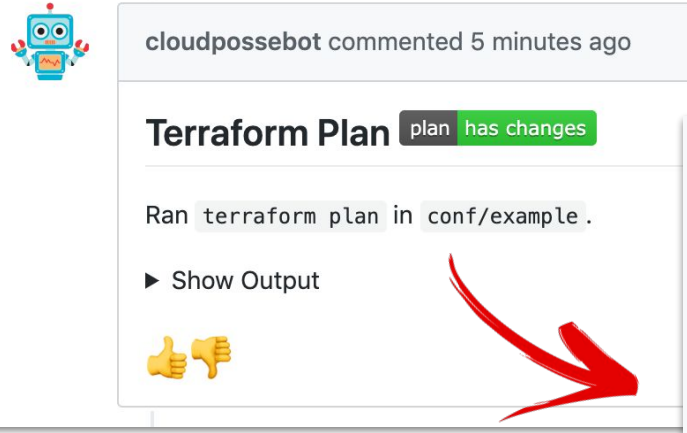
why

- Our application needs to upload artifacts

```
17  + resource "aws_s3_bucket" "default" {
18  +   count  = "1"
19  +   bucket = "codefresh-gitops-example"
20  +   acl    = "private"
21  +
22  +   tags = {
23  +     Name = "Codefresh GitOps Example"
24  +   }
25  + }
```

Step Two: Review “Auto Plan”

What you get




```
+ aws_s3_bucket.default
    id:                          <computed>
    acceleration_status:         <computed>
    acl:                         "private"
    arn:                         <computed>
    bucket:                      "codefresh-gitops-example"
    bucket_domain_name:          <computed>
    bucket_regional_domain_name: <computed>
    force_destroy:               "false"
    hosted_zone_id:              <computed>
    region:                      <computed>
    request_payer:               <computed>
    tags.%:                      "1"
    tags.Name:                   "Codefresh GitOps Example"
    versioning.#:                <computed>
    website_domain:              <computed>
    website_endpoint:            <computed>

Plan: 1 to add, 0 to change, 0 to destroy.
```

Step Three: Seek Approval

osterman

Build is pending approval

Repository
cloudposse/testing.cloudposse.co

Branch
example

SHA
37da8b52

Commit
Add example

Pipeline
example

Approve Deny

slack

CODE REVIEW

Review required
At least 1 approving review is required by reviewers with write access. Learn more.

All checks have passed
1 successful check

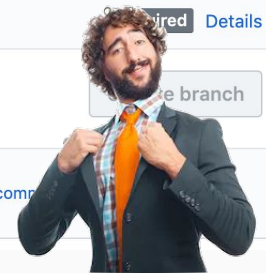
Hide all checks

example — Build passed Details

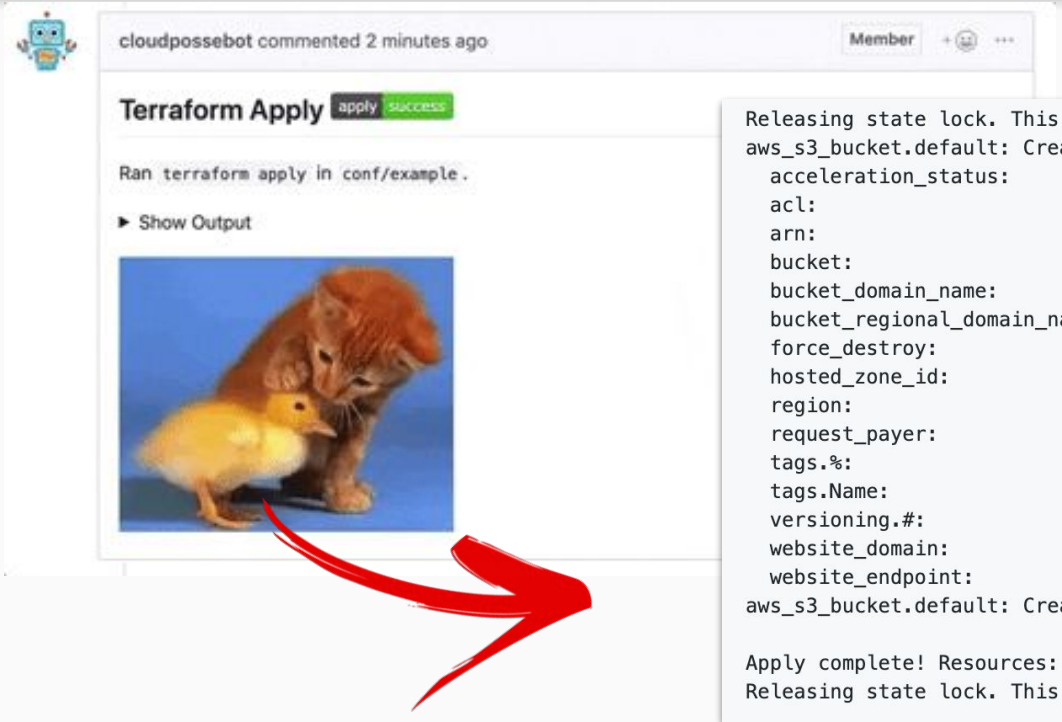
Merging is blocked
Merging can be performed automatically with 1 approving review.

Squash and merge

You can also open this in GitHub Desktop or view comm

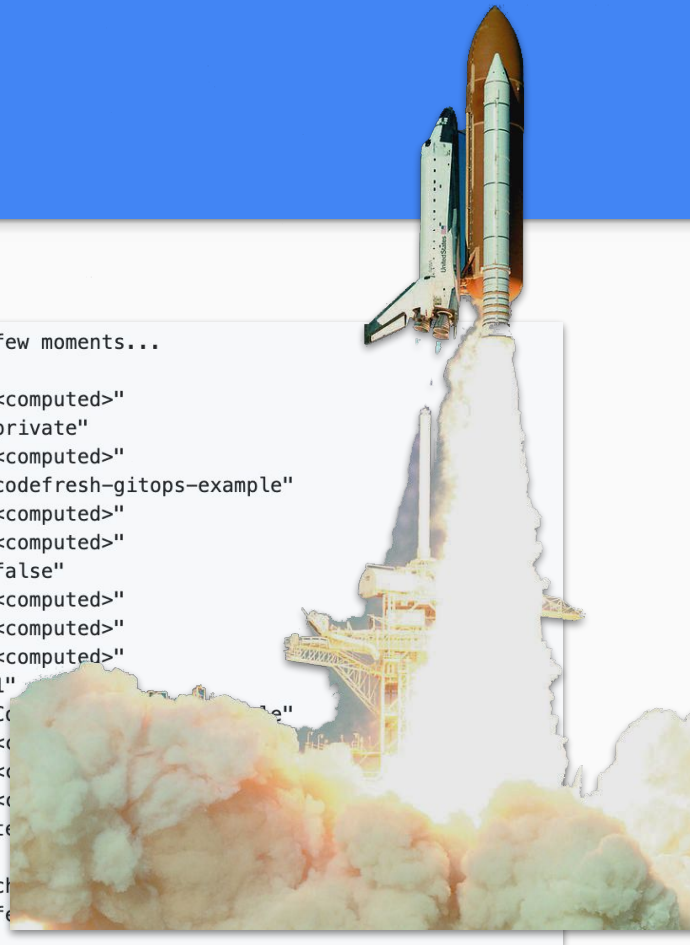


Step Four: Deploy Changes

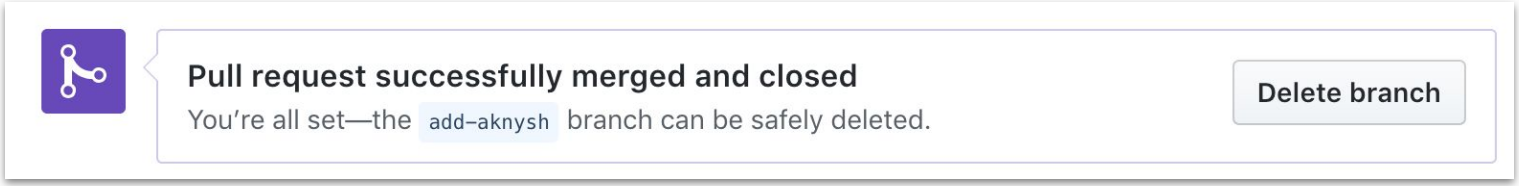



```
Releasing state lock. This may take a few moments...
aws_s3_bucket.default: Creating...
  acceleration_status:         "" => "<computed>"
  acl:                         "" => "private"
  arn:                         "" => "<computed>"
  bucket:                      "" => "codefresh-gitops-example"
  bucket_domain_name:          "" => "<computed>"
  bucket_regional_domain_name: "" => "<computed>"
  force_destroy:               "" => "false"
  hosted_zone_id:              "" => "<computed>"
  region:                      "" => "<computed>"
  request_payer:               "" => "<computed>"
  tags.%:                      "" => "1"
  tags.Name:                   "" => "C
  versioning.#:                "" => "<
  website_domain:              "" => "<
  website_endpoint:            "" => "<
aws_s3_bucket.default: Creation complete

Apply complete! Resources: 1 added, 0 ch
Releasing state lock. This may take a fe
```

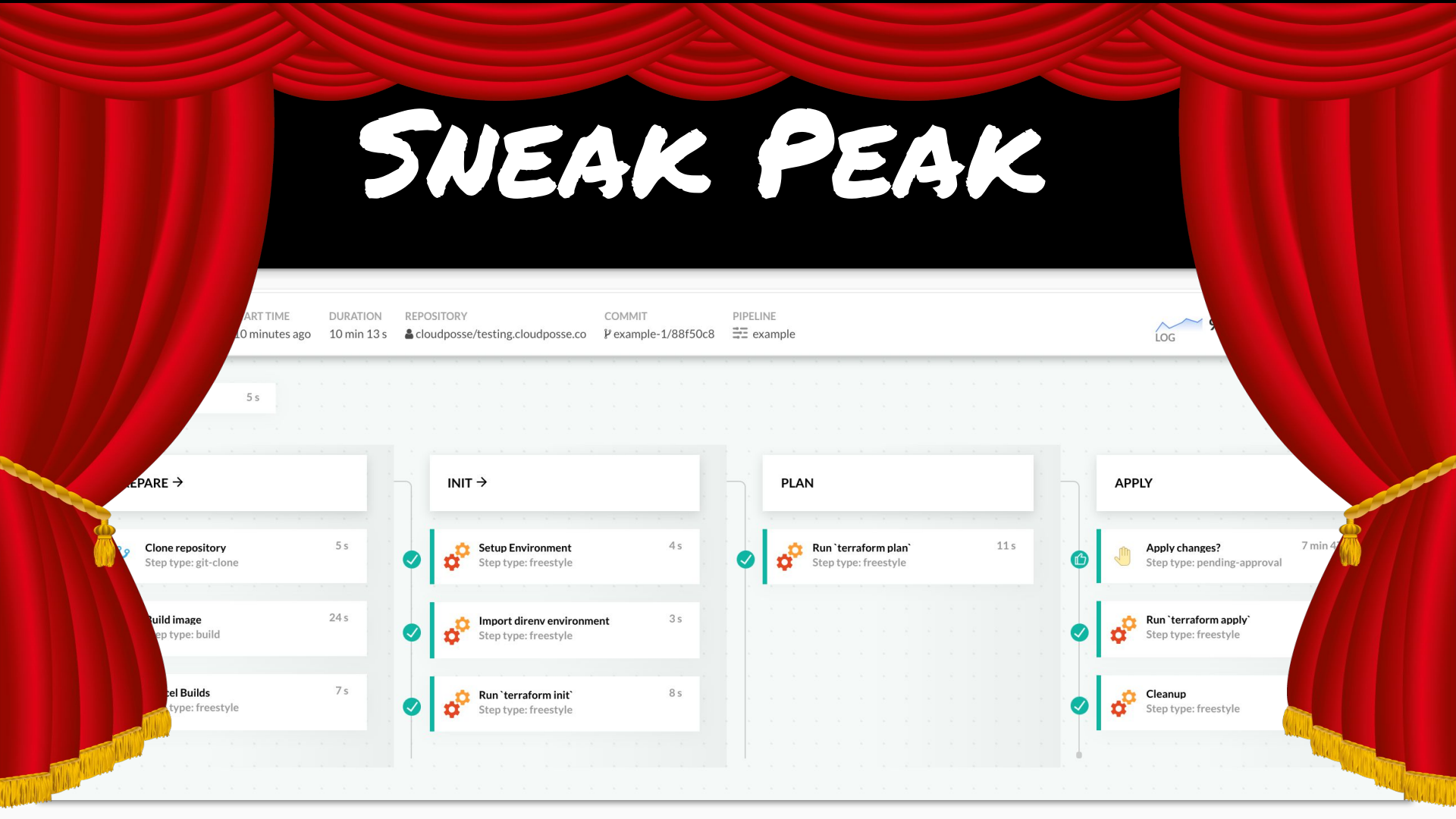


Step Five: Merge Pull Request



Nailed It!

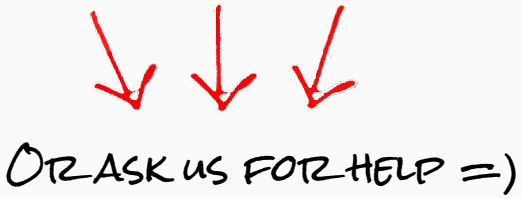
Sneak Peak

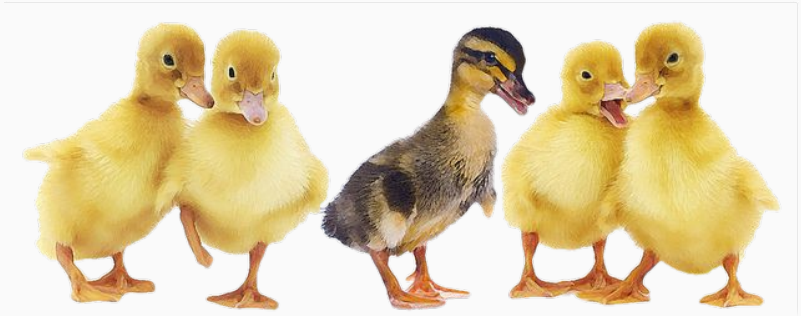

ART TIME
10 minutes ago
DURATION
10 min 13 s
REPOSITORY
cloudposse/testing.cloudposse.co
COMMIT
example-1/88f50c8
PIPELINE
example
LOG
5 s
EPARE →
Clone repository
Step type: git-clone
5 s
uild image
p type: build
24 s
el Builds
type: freestyle
7 s
INIT →
Setup Environment
Step type: freestyle
4 s
Import direnv environment
Step type: freestyle
3 s
Run `terraform init`
Step type: freestyle
8 s
PLAN
Run `terraform plan`
Step type: freestyle
11 s
APPLY
Apply changes?
Step type: pending-approval
7 min 4
Run `terraform apply`
Step type: freestyle
Cleanup
Step type: freestyle

That was **easy**.

How to get started

1. Signup for Codefresh
2. Add `codefresh.yaml` to each terraform repo
3. Get back to work *(sorry it's that easy)*.





Init Step

Example /codefresh.yaml

```
init:
  title: Run `terraform init`
  stage: Init
  fail_fast: true
  image: ${{build_image}}
  working_directory: *cwd
  environment:
  - TF_COMMAND=init
  commands:
  - eval "$(chamber exec atlantis -- sh -c "export -p")"
  - eval "$(ssh-agent)"
  - echo "${ATLANTIS_SSH_PRIVATE_KEY}" | ssh-add -
  - terraform init
```

```
# define step called “init”
# give it a title
# associate it with a stage of the pipeline
# exit on errors
# docker image to use
# working directory (e.g. terraform code)
# environment variables
#   (used for our github comment template)
# commands we should run in this step
#   export environment from chamber to shell
#   start an SSH agent
#   load SSH key so we can pull private repos
#   run terraform init with s3 backend
```

Steps can be Entirely Customized.

Plan Step

Example `codefresh.yaml` (Continued)

```
plan:
  title: Run `terraform plan`
  stage: Plan
  fail_fast: true
  image: ${{build_image}}
  working_directory: *cwd
  environment:
  - TF_COMMAND=plan
  commands:
  - set +e -xo pipefail
  - terraform plan | tfmask | scenery | tee plan.txt
  - export TF_EXIT_CODE=$?
  - github-commenter < plan.txt
  - '[ $TF_EXIT_CODE -ne 1 ]'
```

```
# define step called “init”
# give it a title
# associate it with a stage of the pipeline
# exit on errors
# docker image to use
# working directory (e.g. terraform code)
# environment variables
#   (used for our github comment template)
# commands we should run in this step
# shell flags
# terraform plan, mask secrets, format it
#   record exit code of terraform plan
# comment back to PR with plan output
# exit code of 0 or 2 is success; 1 is error
```

Steps can be Entirely Customized.

Apply Step

Example `codefresh.yaml` (Continued)

```
apply:
  title: Run `terraform apply`
  stage: Apply
  fail_fast: true
  image: ${{build_image}}
  working_directory: *cwd
  environment:
  - TF_COMMAND=apply
  commands:
  - set +e -xo pipefail
  - terraform apply | tfmask | tee apply.txt
  - export TF_EXIT_CODE=$?
  - github-commenter < apply.txt
  - '[ $TF_EXIT_CODE -eq 0 ]'
```

```
# define step called “apply”
# give it a title
# associate it with a stage of the pipeline
# exit on errors
# docker image to use
# working directory (e.g. terraform code)
# environment variables
#   (used for our github comment template)
# commands we should run in this step
# shell flags
# apply the terraform plan and mask output
#     (run apply using previous plan)
#        $PLANFILE ensures WYSIWYG
# Comment back on github with outcome
# Expect an exit code of zero
```

Live Demo

1. ADD USER
2. OPEN PR
3. RUN PLAN
4. SEEK APPROVAL (OR NOT)
5. APPLY
6. MERGE

Demo Time!

Our Best Practices

Fabulous

Use **Geodesic** as our cloud automation shell

Use `IAM STS` for short lived AWS credentials (not hardcoded credentials)

Use GitHub `CODEOWNERS`

Use **.tfvars** for non-secrets

Use **SSM Parameter Store** + KMS for Secrets

Use **scenery** for clean output; **tfmask** to sanitize output

Why do you care?

Teamwork.

GitOps

Stop living dangerously.

Start using GitOps.

- Practice total transparency in operations
- Enable team collaboration
- Reduce access to environments → increase security
- Increase Productivity, Simplify Maintenance, Ensure Repeatability

https://github.com/runatlantis/atlantis

Where can I ask questions?

Join our community!

slack.sweetops.com

Links

SWEETOPS

Example Pipeline on GitHub

cpco.io/codefresh-gitops

github.com/cloudposse/tfmask

github.com/cloudposse/geodesic

github.com/cloudposse/github-commenter

$500/mo - 2 hours

Office Hours with Cloud Posse

Why you want it...

- **Expert Advice** — Prescriptive solutions to your questions
- **Reduced Time to Market** — know your options & eliminate analysis paralysis
- **Trusted Partner** — who learns your stack and understands your problems

100% ZERO RISK

30 DAYS MONEY BACK GUARANTEE

VISA MasterCard PayPal

What you get...

- **Recorded Strategy Sessions** — Weekly or Biweekly Cadence (30m-1hr)
- **Easy Scheduling** — via Calendly or recurring events
- **Shared Slack Channel** — for private communications (~12 hour SLA)

Cloud Posse

A Totally Sweet DevOps Professional Services Company

Hire us. =)

100+ Free Terraform Modules	github.com/cloudposse
Active Community	sweetops.com/slack
Awesome Documentation	docs.cloudposse.com

(**free** consultation)