Cooking Infrastructure isn’t illegal

aka. How not to Break things Bad

Startup Safari 2018

CHEPPERS

Agenda

- Introductions
- Bad Infrastructure
- How Not To Cook
- Enter AWS
- CloudFormation
- Managing CloudFormation stacks - Furnace
- Live Coding
- Why Furnace

Introductions

CHEPPERS

Bad Infrastructure

CHEPPERS

Common Patterns

- Un-reproducable
- Fragile
- Difficult to maintaine
- Has leprechauns (hand made changes)
- Too many "Why is this here? What does it do?" moments

CHEPPERS

How not to cook

Gordon Ramsay would be sad

- Using various recipes (have puppet and chef and ansible at the same time)
- Overcook (you infrastructure code is too complicated)
- Though, that's how pizza was made (have bread and flavour it with whatever you can find)

Enter AWS

Little History

Basement Servers

aka. the great sysadmin wars of 2001

https://i.ytimg.com/vi/SVRT5TTr-Bk/maxresdefault.jpg

You don’t need these unless you really really REALLY REEEAALLLYYY want your data locally / you’re a bank

Using AWS

Bad usage

- Having a single, monolith EC2 instance with
- No VPC
- No Security Groups
- No Configuration management to manage it

Might as well stick to Linode / Digital Ocean at that point.

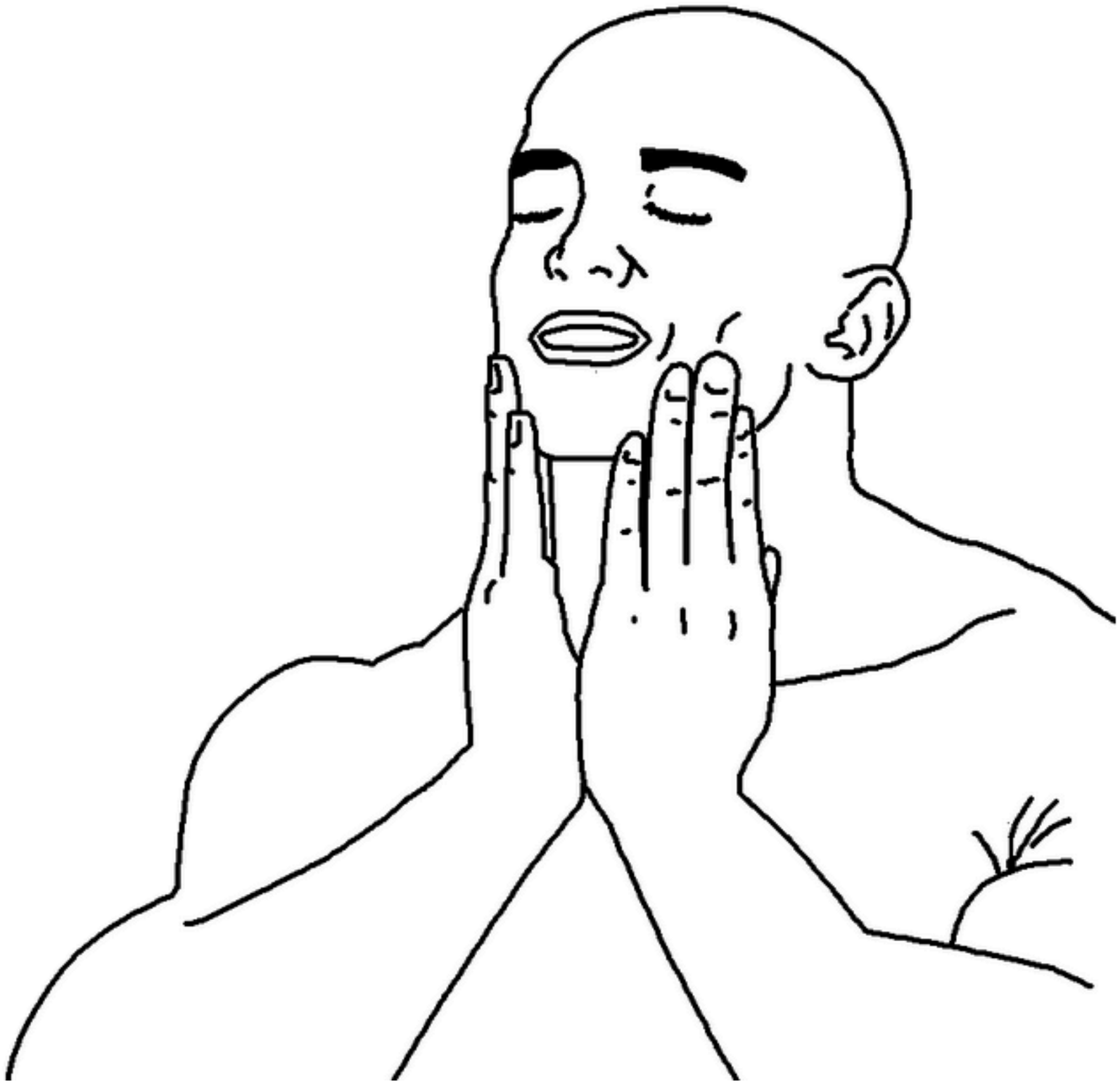
Better usage

- Still, single EC2, but it's managed by a configuration management system
- Ansible, Chef, Puppet, Salt...
- But...
- No VPC
- No Security Groups
- No real single view of how many components there are

Good usage

- Security Groups and VPC applied
- Optionally: LoadBalancers, Autoscaling Groups
- Things are composed through some kind of configuration management
- Still... Things might be hand crafted / created. For example: Security groups, IAMs, VPCs or other AWS resources

Perfection...



CloudFormation

https://aws.amazon.com/cloudformation/

Idenpotent

- What is CloudFormation?
- How can it help?
- Why would I use it?

Managing CloudFormation stacks - Furnace

https://github.com/Skarlso/go-furnace

Configuring Furnace

```
main:
  stackname: FurnaceStack
  spinner: 1
aws:
  code_deploy_role: CodeDeployServiceRole
  region: us-east-1
  enable_plugin_system: false
  template_name: cloud_formation.template
  app_name: furnace_app
code_deploy:
  # Only needed in case S3 is used for code deployment
  code_deploy_s3_bucket: furnace_code_bucket
  # The name of the zip file in case it's on a bucket
  code_deploy_s3_key: furnace_deploy_app
  # In case a Git Repository is used for the application, define these two settings
  git_account: Skarlso/furnace-codedeploy-app
  git_revision: b89451234...
```

Configuring a Stack

`stacks/mystack.yaml`

Create Stacks with Furnace

```
faws create
```

Deploy Applications with Furnace

```
faws push
```

Using Multiple Furnace files

```
- webstack
|-- webstack.yaml
|-- webstack.template
|
- databasestack
|-- database_stack.yaml
|-- database_stack.template
|
- .webstack.furnace
- .dbstack.furnace
```

Creating both stacks

```
faws create webstack
faws create dbstack
```

Live coding

Commence some magic here

In case live coding fails...

Output of Create

```
[/] Waiting for state: CREATE_COMPLETE
... Post-create plugin events...
... Stack state is: CREATE_COMPLETE
```

Available maintainance commands

```
create
update
status
delete
push
delete-application
```

Why Furnace?

* Single binary
* No dependencies
* Dead simple to use

This isn’t about Furnace

Thank you!

https://github.com/Skarlso/go-furnace