Infrastructure as Code (IaC) Workshop - Practice

By Sven Balnojan

The What

In this "workshop", I introduce a bunch of tools

- the AWS CLI
- Terraform (& possibly Pulumi)

And a list of concepts & principles including:

- Task Group 1: Reproducibility (& making scripts reproducible)
- Task Group 2: TDD for IaC, Declarative Code, IaC State, Terraforms Rule No. 1 (& mocking, local unit testing IaC)
- Task Group 3: Intro TF.
- Task Group 4: Disposability, Continual Availability. (& Modular programming is great!)
- Task Group 5: High Level Programming Languages & their uses in IaC.

How I like to "teach tech":

- Learning the macro from the micro. (That means, we start as small as possible and learn principles which then can be transferred to everything else. It also means the exercises sometimes use a "wrong" tool to teach you something, a tool/solution you shouldn't use like that in production.)
- Test-driven to the core (Thanks Uwe Schäfer for helping me internalize that!)
- ./go is the way to go.

Please choose a partner! (On exercises call each other, and watch the timer!) If in doubt, call me :-)

Task Group 1 - Your Turn

Tool: AWS CLI

Exercise 1: Create a S3 bucket, non-public access, with your name.

Task Group 1 - Your Turn

Tool: AWS CLI

Exercise 2: Ok great, now take a look at what you did. Do you think someone else could reproduce that? Make it reproducible with the minimal amount of effort.

Task Group 1 - Your Turn

Tool: AWS CLI

Exercise 3: Nice, now take a final look at what you created and see whether it adhere's to good practices. Does someone else has to edit it to run it as well with e.g. a different name? Adjust that.

Task Group 1

Principle: Reproducibility. (Making scripts reproducible & reuseable, parametrizing,...)

Side Note: For bash I recommend ShellCheck & the Google & Gruntworks styleguide (https://github.com/gruntwork-io/bash-commons)

Task Group 2 - Question + Demo

QUESTION: Why do we need a “stub” or a “mock” NOW? (And what’s the difference?)

Demo: Running local AWS tests

Demo: To test AWS resources locally you got to redirect the "endpoint". This can be done like this:

Demo: Bash tests

Demo: Let’s write a test for what we just created together...

Task Group 2 - Questions for You

Question: Now run the test a second time... What happens?

Task Group 2 - Questions for You

Question: Now run the test a second time... What happens?

(Uh oh, new concept: Declarative vs. Imperative code...)

Question 2: What if we were to randomize the name?

Task Group 2 - Answer

Question: Now run the test a second time... What happens?

(Uh oh, new concept: Declarative vs. Imperative code...)

Question 2: What if we were to randomize the name?

Both problems could be handled with by:

Doing a "if is already there, do nothing otherwise create it" logic (a deployment mechanism)

Saving the name of the bucket to file (bucket.state)

Combining both things...

And that is, what we have the concepts: Declarative Code & State of our infrastructure for. It also displays the rule no. 1 of terraform.

Task Group 3 Demo

Short TF demo together... (run the test, it fails! We fix it together...)

1. Terraform init + providers & local mocking...
2. Files + variables (looks nice to work with right?)
3. Terraform plan, (tfstate file)
4. Fix...

Task Group 4 Your Turn!

Let's now take care of one important principle, "disposability".

(For that we got to switch away from local testing though for now..)

...

====

$ exercises/libs/bats/bin/bats exercises/exercise_4.bats

"Create a EC2 Instance with an apache webserver"

... ERROR

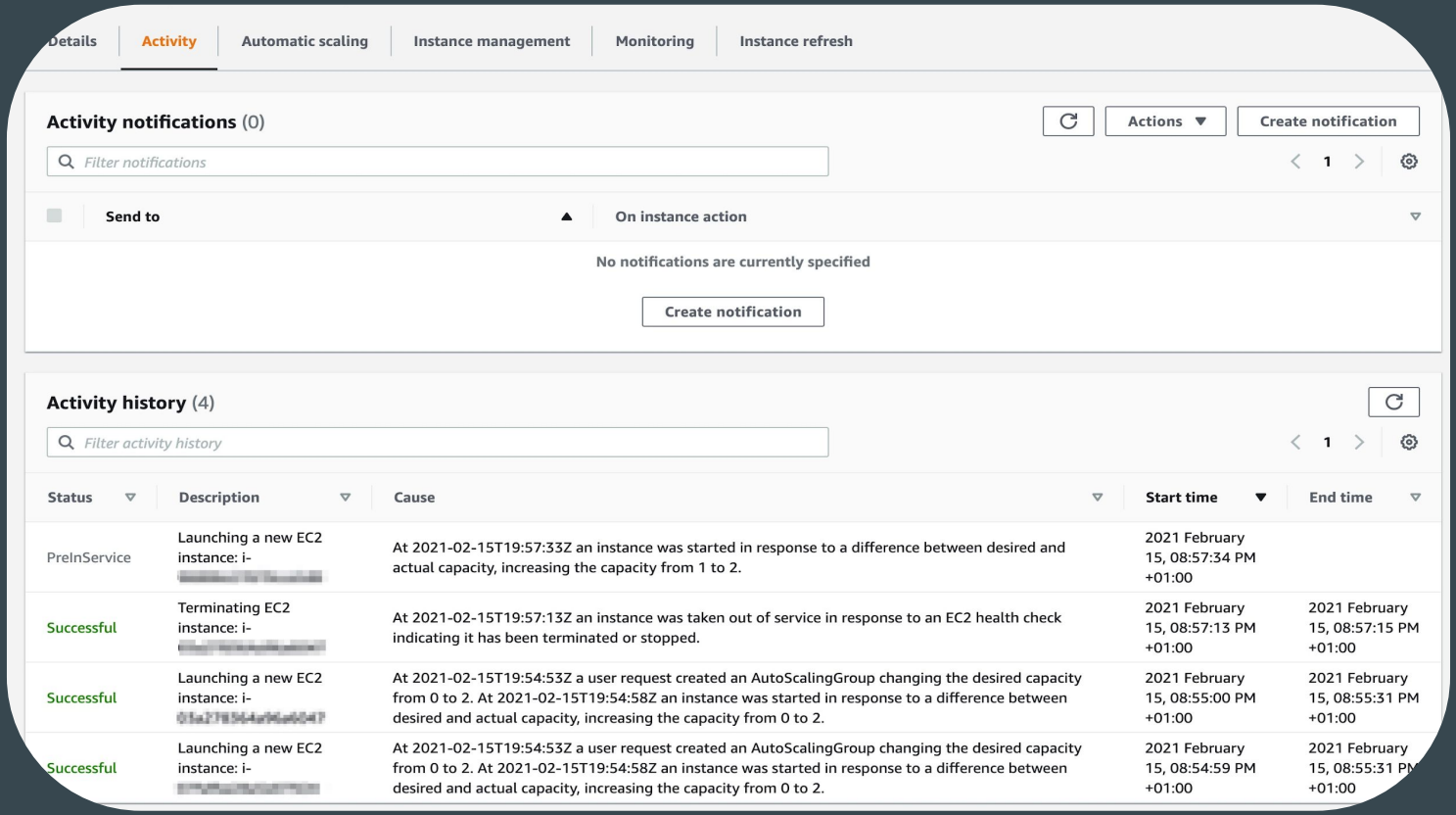
====

Task 1: Fix this...

Task Group 4 Your Turn

Task 4.2: Now let's make this "disposable". Terminate the instance. Does it restart? Nope... so let's take care of that (hint: you'll need an ASG...MAYBE you can find a module...)

Task Group 4 Demo Terminate



Task Group 5 Demo

Great, so we got TF with it's custom DSL. Now let's look at Pulumi in Python.

... (pulumi-example)[pulumi-example]

... Check out how to set up our local dev environment again, by overriding the endpoint (get's boring doesn't it?)

... setting up python venv stuff:...

...Pulumi up

...Pulumi destroy

Task Group 5 Your Turn

Exercise 5: We want to experience a slight sight of the greatness of high level programming languages right? So let's for now simply create 10 similar buckets, all private, but with a differing name. (Hint: use functions & control constructs)

Thanks! Enjoy!