InteropITX
APRIL 30 – MAY 4, 2018
THE MIRAGE | LAS VEGAS

Infrastructure as Code for Beginners

A hands-on workshop for getting started with infrastructure as code

Scott Lowe
Staff Field Engineer
Heptio

Before we begin

- Get involved! Audience participation is requested
- If you use social media, feel free to post updates about this presentation (on Twitter include the **#InteropITX** hashtag or **@Interop** handle)
- You are welcome to take photos or videos of today's presentation and share them online
- This presentation is available for download from the Interop web site

A quick intro

- Husband, father, Jeeper, all-around geek
- Blogger (almost 13 years at https://blog.scottlowe.org)
- Author (8 books so far)
- Speaker (Interop, VMworld, DevOps Networking Forum, OpenStack Summit, local meetups)
- Podcaster (The Full Stack Journey podcast)
- Staff Field Engineer at Heptio

What we'll cover in this workshop

- Basics of infrastructure as code
- A quick introduction to Git for version control
- Using Ansible for infrastructure as code
- Using Terraform for infrastructure as code

Basics of infrastructure as code

Basics of infrastructure as code

- Infrastructure as code is “an approach to infrastructure automation based on practices from software development” (taken from O’Reilly’s “Infrastructure as Code,” by Kief Morris)
- Some specifics include:
  - Using a version control system (VCS) to store automation artifacts
  - Incorporating various automation tools
  - Leveraging testing and validation mechanisms and tools
  - Thinking “programmatically” about how to use infrastructure automation tools

Some common aspects of infrastructure as code

- Creation of server templates
- Orchestration of infrastructure elements or IaaS platforms (to create network, security, storage, etc., constructs)
- Deployment of servers onto IaaS platforms
- Configuring/update deployed servers
- Making configuration changes to infrastructure or IaaS elements

Some common tools used in infrastructure as code

- Git (and possibly GitHub or GitLab)
- Configuration management tools (Ansible, Chef, Puppet, Salt)
- IaaS orchestrators (CloudFormation, Azure Resource Manager, Terraform)
- OS image creation tools (Packer, Aminator)
- IaaS-specific tools (AWS CLI, Azure CLI, GCP **`gcloud`** tool)

What we'll do with infrastructure as code in this workshop

We'll look at a few examples of infrastructure as code:

- Create some AWS infrastructure using Terraform
- Use Ansible to configure the EC2 instances created by Terraform

What questions do you have so far?

A quick introduction to Git

Brief history and background on Git

- Created by Linus Torvalds in April 2005
  - Was self-hosting within a few days (Git was managing Git)
  - Managed the 2.6.12 kernel release within a couple months
  - Git 1.0 released at the end of 2005
- Key design goals: fast, simple, scalable, distributed, strong support for non-linear development (branching)
- Latest version (as of March 2018) was 2.16.2 (for details see https://git-scm.com)

Some Git terminology

- **Repository:** The database that contains all of a project's information and history. Once added to the repository, information is immutable
- **Working directory:** The area where the user can modify the files in the repository before committing them
- **Index:** A binary file maintained by Git that describes the repository's directory structure and content at a point in time
- **Bare repository:** A repository without a working directory
- **Commit:** An entry in the repository (Git database) recording metadata for each change made to the repository
- **Remote:** A link to another Git repository

Very basic Git workflow

1. Run **`git pull`** to retrieve latest changes from a remote repository (when applicable).
2. Create or modify files in the working directory.
3. Run **`git add`** to stage new or modified files.
4. Run **`git commit`** to commit the staged changes.
5. Run **`git push`** to push changes to a remote repository (when applicable).

Let's review some Git concepts in action!

Git demo and hands-on lab

Prerequisites: A GitHub account and Git installed on your system

Lab instructions at https://github.com/scottslowe/2018-itx-iac-workshop

What questions do you have about Git?

Using Terraform for infrastructure as code

Quick overview of Terraform

- Designed to build, modify, and version infrastructure
- Supports a variety of services/platforms via providers (AWS, Azure, GCP, vSphere, Kubernetes, etc.)
- Can orchestrate multiple platforms/providers in a single configuration
- Written in Go by HashiCorp, is an open source project

Let’s review some Terraform concepts and code!

Terraform demo and hands-on lab

Prerequisites: An AWS account and Terraform installed on your system

Lab instructions at https://github.com/scottslowe/2018-itx-iac-workshop

What questions do you have about Terraform?

Using Ansible for infrastructure as code

Quick overview of Ansible

- Designed to execute tasks on remote systems
- Tasks can be chained together in playbooks for multi-step orchestration
- Written in Python and leverages SSH
- Most modules (which extend Ansible's functionality) are idempotent
- Although mostly used for configuration management, can be used for IaaS orchestration as well

Let’s look at some Ansible concepts and code!

Ansible demo and hands-on lab

Prerequisites: An AWS account and Ansible installed on your system

Lab instructions at https://github.com/scottslowe/2018-itx-iac-workshop

What questions do you have about Ansible?

In summary

- We discussed the basic concepts behind infrastructure as code
- We reviewed some of the common aspects of infrastructure as code
- We examined some common tools practitioners use when using infrastructure as code
- We saw some examples of infrastructure as code in action

OK, last chance: what questions do you have?

Thanks for attending!

Please be sure to fill out the session survey and provide feedback!