Carnegie Mellon

Deployment and Operations for Software Engineers 2nd Ed

Chapter 10 – Basic DevOps tools

Outline

- **Infrastructure as Code**
- Issue Tracking
- Version Control
- Provisioning and Configuration Management
- Vendor Lock in
- Configuration parameters

Infrastructure as Code

- Infrastructure as Code (IaC) is the management of infrastructure (networks, virtual machines, containers load balancers, and connection topology) using code segments in various languages. E.g.
  - Command line scripting
  - Provisioning specifications, e.g. Vagrant, Cloud Formation, Chef, Puppet, Ansible
  - Specification files for various tools, e.g Dockerfile

Why IaC?

- Having a script for repeated operations has the following advantages
  - Actions can be invoked simply
  - Reduces errors due to human mistakes during data entry
  - Allows different team members to share scripts
  - Makes infrastructure uniform and reproducible
- Best practices
  - Scripts are checked into version control system
  - Scripts are scanned for security issues
  - Scripts are tested and reviewed for correctness, security, and compliance

Costs of IaC

- Scripts must be developed
- Scripts must be tested and reviewed
- Scripts must be updated when tools or processes are modified
  - Must be tested, reviewed
  - Loads to longer turnaround times

Idempotence

- Something is *idempotent* if applying it twice yields the same result. E.g. identity function.
- *Idempotence* is a principle of Infrastructure as Code - a deployment command always sets the target environment into the same
  configuration, regardless of the environment's starting state.
- Idempotence is achieved by either automatically configuring an existing target or by discarding the existing target and recreating a fresh environment.
- IaC scripts are typically idempotent

Infrastructure drift

- Suppose you've deployed five instances via IaC with the default 128 MB of memory,
- Now three of them are using 256 MB
- How did this happen?
  - Someone noticed instances had poor performance and increased the memory allocation
- Is this bad?
  - Suppose poor performance is noticed on the five instances. Is it one of the smaller memory instances?
  - It makes it more difficult to find problem.

Secuity

- Infrastructure drift can also cause security problems.
- Someone is having problem and changes the IAM setting to make access easier.
- Depending on how it was changed, it may open up a system to unauthorized usage.

Detecting infrastructure drift

- Several open source tools can check for infrastructure drift.
  - Snyk
  - Driftctl
  - …
- During production

Discussion questions

1. How would you decide that writing a script was worth while from a cost perspective?
2. You have written a script for some purpose. How do you get your team or organization to adopt it? What would that mean in terms of other roles within your organization?
3. Why does a declarative language make idempotence easier?

Outline

- Infrastructure as Code
- **Issue Tracking**
- Version Control
- Provisioning and Configuration Management
- Vendor Lock in
- Configuration parameters

Issue tracking

- An issue can be a bug, a desired feature, or an incident.
- An issue is entered into the tracking database and given an ID. That ID is then used to identify subsequent activities.
- IDs are the key for tying together activities ranging from code development to the steps in the deployment pipeline to incidents that occur during operations.
- Issue trackers are not, *per se*, DevOps tools since they date from before DevOps became a movement.
- Common issue trackers are Issue Tracker, Jira, and Solar Winds.

Outline

- Infrastructure as Code
- Issue Tracking
- **Version Control**
- Provisioning and Configuration Management
- Vendor Lock in
- Configuration parameters

Version Control

- A version control system (VCS) keeps track of modifications to textual files.
- The textual files can be programming language code, scripts, or any other system specific file such as documentation.
- The VCS stores the files and their modifications in a repository.
- The repository is shared among team members.
- VCS has access controls.

Basic VC commands

- Check in/check out. Check out will copy a branch or a file from the repository to the local file system. Check in reverses the process.
- Branch/merge. A branch creates a new copy of the set of files being branched. Merge identifies the differences between the two branches being merged and gives the option of choosing one alternative to continue in the merged branch.
- Version labeling/tagging. Versions can be labelled automatically or manually

Centralized VCS

- The end user must be online to check a file in or out.
- The VCS keeps track of which user has checked out a file and can prevent other users from checking out the file until it has been checked in.
- Subversion is a widely used centralized VCS.

Distributed VC

- The end user makes a local copy of either the whole repository or a branch.
- The user must be online to make this local copy.
- Once the local copy has been made, check in and check out of individual files can be performed without an internet connection.
- Git is a widely used decentralized VCS.

Common Branching strategies

- Centralized workflow: Teams use only a single repository and commit directly to the main branch.
- Feature branching: Teams use a new branch for each feature and don't commit directly to the main branch.
- Personal branching: Similar to feature branching, but rather than develop on a branch per feature, it's per developer. Every user merges to the main branch when they complete their work.

One repo or several repos?

- Organizations must decide whether to use one repository for all their projects or project specific repositories.
- A single repository facilitates reuse since all code is available to all developers. It also allows any developer to fix a problem with any code developed by the organization.
- Multiple repositories decouple one project from another. The developers on a project operate independently in terms of libraries, development processes, and merging branches.

Best Practices

- Use descriptive comments when committing and identify the commit with an ID from an issue tracker.
- Do not commit incomplete code.
- Commit logical units. If you are simultaneously working on two different issues, then commit them separately.
- When an issue has been resolved, commit that code rather than waiting until all of the issues on which you are working are resolved.

Security

- In order to avoid insider attacks, some organizations require two people to verify a check in.
- A security review of the code must precede any check in.
- An insider attack would require two people to collaborate on the attack. This is much less likely than a single individual mounting an attack.
- A side effect of requiring a security review is that code quality will be improved, as it is with any review.

Outline

- Infrastructure as Code
- Issue Tracking
- Version Control
- **Provisioning and Configuration Management**
- Vendor Lock in
- Configuration parameters

Provisioning tools

- Provisioning tools either create or modify
  - A node or container at the target
  - An environment

Considerations for provisioning tools

- A provider is included in the specification either implicitly or explicitly.
- The specification will include a target name. If the target already exists, then executing the specification will make this target consistent with the specification. If the target does not exist, executing the specification will create the target.
- The desired software is loaded from the provider locations specified.
- You must have permissions to access the target provider, an existing node, and the desired software.

Common provisioning tools

- Docker. Docker creates and provisions containers.
- Vagrant. Vagrant creates and provisions development environments.
- Terraform. Terraform creates and manages infrastructure. For example, your environment might consist of three instances of your system each running on a host, managed by a load balancer, and connected to the internet.
- Cloud Formation. This is an AWS specific provisioning tool. It enables you to create a VM within AWS. You specify the instance type, the image to be loaded into the instance, the security group settings, etc.

Configuration Management tools

- A configuration management (CM) tool copies a master copy of a set of software to a collection of specified nodes
- A provisioning tool manages a single node or environment
- A configuration management tool manages a collection of instances and keeps them consistent. It prevents infrastructure drift.

CM operations

- Instances are categorized and the CM tool will manage the nodes of the collection. Sample categories
  - Web servers
  - Developer' laptops
- Typically, the configuration management tool executes on one node, the files to be copied exist on a file master and SSH or TLS with self- signed certificates is used to connect to the target node and perform the copying

Common CM tools

- Chef. https://www.chef.io/products/chef-infrastructure-management
- Puppet. https://puppet.com/
- Ansible. https://www.ansible.com/

Discussion questions

1. What are some security considerations for provisioning tools?
2. What are some security considerations for configuration management tools?

Outline

- Infrastructure as Code
- Issue Tracking
- Version Control
- Provisioning and Configuration Management
- **Vendor Lock in**
- Configuration parameters

Vendor lock in

- Vendor lock in occurs when you are dependent on a single vendor for a tool or services and have no economical method to switch to a different vendor.
- Some tools are vendor pecific. E.g. Cloud Formation only works with AWS
- Some tools are vendor agnostic. E.g. Vagrant has a provider field that specifies the target. Targets include VirtualBox, Hyper-V, Docker
- Some tools are portable. E.g. Chef, Puppet, and Ansible all can operate on different cloud providers.

Outline

- Infrastructure as Code
- Issue Tracking
- Version Control
- Provisioning and Configuration Management
- Vendor Lock in
- **Configuration parameters**

Configuration parameters

- A configuration parameter is a parameter intended to be specified by a system administrator.
- Despite the similarity in names, there is no connection between configuration parameters and configuration management tools.
- Examples of configuration parameters include network information (e.g., where is your DNS server?), database-connection information, logging levels, user interface background color, localization information, and security levels etc.

Why configuration parameters

- A developer decides that a parameter is to be configuration parameter when
  - It is unreasonable to ask an end user to set it
  - The value of the parameter may change in different environments
- Configuration parameters should
  - Be checked for reasonableness
  - Have default values
  - Be version controlled

Methods for setting configuration parameters

- *Resource file.* A resource file is a file with a name and location known to the service. The service will read the file at initialization and assign the configuration parameters. *Environment variables.*
- An *environment variable* is a variable in the operating system
- *Database.* A specialized database can be used to store configuration parameters.
- *Specialized tools.* Specialized tools provide the ability to interactively specify configuration parameters.

Security issue

- Configuration-parameter names and values are typically stored in cleartext. They are not hidden or encrypted, and they are visible to anyone who has access to the source code for your infrastructure.
- DO NOT put credentials in scripts.
  - Scripts are stored in version control systems available to many users
  - Public repositories such as GitHub are scanned by malicious actors looking for credentials.
- Use a tool such as Vault (discussed later) to manage credentials.

Discussion questions

1. What are the tradeoffs in the different methods for setting configuration parameters?
2. Why would you make background color a configuration parameter?