Carnegie Mellon

DevOps-Continuous Integration

Baisic DevOps tools

1

Carnegie Mellon University

Overview

- **Version Control**
- Provisioning tools
- Configuration Management Tools
- Configuration Parameters
- Managing Secrets

2

Carnegie Mellon University

Version Control Systems (VCS)

- Maintains textual information
- Shared among team members
- Centralized or distributed
- Three functions
  1. A check-out/check-in process
  2. A branch/merge process
  3. Tagging or labeling versions

Carnegie Mellon University

Centralized version control system

- A centralized VCS has a central repository.
- Users must connect to that central repository to check in or check out files. I.e. internet connection is required.
- System can maintain knowledge of who is working on which files.
  - Allows informing team members if another member checks out a file
  - Allows locking of files or locking of check in.
- Subversion is common centralized VCS.

Carnegie Mellon University

Distributed VCS

- A distributed VCS also has a central repository but interactions are different from centralized VCS
- User gets a copy of the repository for local machine
- Check in/check out of a file is from local copy.
- Does not require internet connection to check out
- System has no knowledge of which team member is working on which file.
- Git is common distributed VCS.

Carnegie Mellon University

Check in/Check out

- Check out results in a local copy
- User can modify local copy.
- Check in copies it back to repository – local or central.
- New version gets new number.
- VCS can perform style checks on check-in

Carnegie Mellon University

Branch process

- Repository is structured as a tree with branches
- Files exist on a branch
- New branches can be created with a duplicate set of files. Allows for different tasks on same file. E.g. adding features and performing bug fixes. These actions might be done on different branches.
- Each branch has a purpose and the files on one branch are isolated from the files on a different branch.

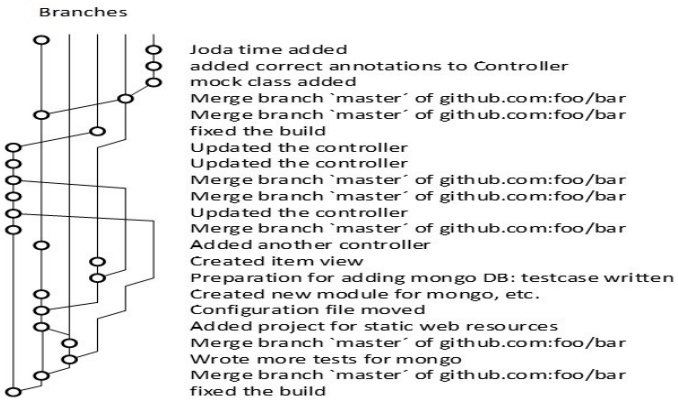
Carnegie Mellon University

Merge process

- Two branches can be merged.
- Differences in two versions of the same file must be resolved.
- Best practice is to merge frequently since reduces number of differences that must be resolved.

Carnegie Mellon University

Sample branch structure



1

institute for SOFTWARE RESEARCH

9

Carnegie Mellon University

Best practices for version control

- Use a descriptive commit message
- Make each commit a logical unit
- Avoid indiscriminate commits
- Incorporate others' changes frequently
- Share your changes frequently
- Coordinate with your co-workers
- Remember that the tools are line-based
- Don't commit generated files

institute for SOFTWARE RESEARCH

10

Carnegie Mellon University

Overview

- Version Control
- **Provisioning tools**
- Configuration Management Tools
- Configuration Parameters
- Managing Secrets

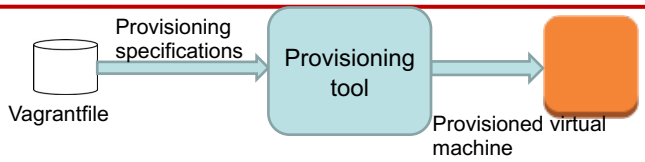
Carnegie Mellon University

Consistency is important

- Consistency on platforms reduces error possibilities.
  - Different team members should develop on the same platform
  - Development platform should be consistent with the production platform
  - Updates should be applied to all machines in a fleet automatically
  - Replicas across data centers should have same software installed down to version and patch numbers.

Carnegie Mellon University

Provisioning tool



- Provisioning tool takes a specification for a provisioned VM and produces such a VM on a provider such as VirtualBox
- Specification is version controlled and shared among team members through version control system
- Result is a common environment for development and production

Carnegie Mellon University

Common provisioning tools

- Vagrant – can have targets of Virtual box, AWS, VMWare, etc
- Cloud Formation – AWS specific
- Azure Resource Manager – Azure Specific

Carnegie Mellon University

Overview

- Version Control
- Provisioning tools
- **Configuration Management Tools**
- Configuration Parameters
- Managing Secrets

Carnegie Mellon University

Configuration Management (CM) Systems

- A CM system is essentially a copier. It copies files from one server to a variety of other servers.
- The purpose of a CM system is to maintain consistency across a set of machines.
- Common tools are:
  - Chef
  - Puppet
  - Ansible

Carnegie Mellon University

Actions of a CM tool

- A CM tool resides on a server
- Controls collection of other machines – virtual or physical
  - Identified by IP addresses and grouped by functions. E.g. Web server, Development platform, DB server
  - Copies them through SSH (mostly) from CM server to clients
- Extensible through plug-ins

Carnegie Mellon University

Specification

- Specification of actions is done through scripting language
- E.g. “make sure all web servers have latest patch for nginx 15.8”
- Specification should be version controlled.

Carnegie Mellon University

Overview

- Version Control
- Provisioning tools
- Configuration Management Tools
- **Configuration Parameters**
- Managing Secrets

Carnegie Mellon University

Configuration parameters

- A configuration parameter is a parameter set by system administrator
- . E.g., logging level, DB URL, background color
- As service moves from development to production, different values will be used for configuration parameters.

Carnegie Mellon University

Management of configuration parameters

- Incorrect parameters are a major source of errors
- Libraries exist for some types of checking. E.g.
  - Values have been specified
  - Range of value is correct
- URLs for external entities should be checked during startup to ensure
  - They are accessible from current environment
  - They are of the correct type.
- Need to worry about maintaining confidentiality of secret configuration parameters.

Carnegie Mellon University

Getting parameter values to application

- Resource file. Read by app, typically on initialization
- Environment variables. Set in operating system and used by name in app.
- Database. Read by app.
- Specialized tool. Used prior to invocation to set parameter values.

Carnegie Mellon University

What parameters should be configuration parameters?

- Developers decide which parameters are configuration parameters.
- Form of late binding.
- Any parameter can be made a configuration parameter
  - Must have default value if not set
  - Value must be checked for reasonablenesss
  - Possible to have too many configuration parameters, e.g. >10,000 in some systems.

Carnegie Mellon University

Overview

- Version Control
- Provisioning tools
- Configuration Management Tools
- Configuration Parameters
- **Managing Secrets**

Carnegie Mellon University

Credentials as configuration parameters?

- It is tempting to make credentials, e.g. passwords, keys, be configuration parameters
- Making a credential a configuration parameters will expose secrets too broadly.
- Configuration parameters show up in textual scripts. Means anyone with access to script has access to credentials. May be stored in public repository such as Github.

Carnegie Mellon University

Managing credentials

- Credentials should be treated differently from other parameters.
- Credential management covered in Applied Distributed Systems
  https://youtu.be/cvxAOp5Hl1g

Carnegie Mellon University

Summary

- Version control systems enable controlled sharing and modification
- Provisioning tools create an environment
- Configuration management systems maintain consistency of software and versions across classes of machines
- Configuration parameters are a form of deferred binding.
- Credentials should not be made configuration parameters.