Configuration management

How configuration management helps engineering teams build robust and stable systems

BY IAN BUCHANAN

During the 1950s the United States Department of Defense developed a technical management discipline to track changes in the development of complex systems. It gave this system and various iterations very technical names, until in 2001 it published a consolidated guidebook that established the technical management system now called configuration management. Today, configuration management is not only used by the defense department, but in software development, IT service management, civil engineering, industrial engineering, and more.

What is configuration management?

Configuration management is a systems engineering process for establishing consistency of a product's attributes throughout its life. In the technology world, configuration management is an IT management process that tracks individual configuration items of an IT system. IT systems are composed of IT assets that vary in granularity. An IT asset may represent a piece of software, or a server, or a cluster of servers. The following focuses on configuration management as it directly applies to IT software assets and software asset CI/CD.

Software configuration management is a systems engineering process that tracks and monitors changes to a software systems configuration metadata. In software development, configuration management is commonly used alongside version control and CI/CD infrastructure. This post focuses on its modern application and use in agile CI/CD software environments.

Why is configuration management important?

Configuration management helps engineering teams build robust and stable systems through the use of tools that automatically manage and monitor updates to configuration data. Complex software systems are composed of components that differ in granularity of size and complexity. For a more concrete example consider a microservice architecture. Each service in a microservice architecture uses configuration metadata to register itself and initialize. Some examples of software configuration metadata are:

- Specifications of computational hardware resource allocations for CPU, RAM, etc.
- Endpoints that specify external connections to other services, databases, or domains
- Secrets like passwords and encryption keys

It's easy for these configuration values to become an afterthought, leading to the configuration to become disorganized and scattered. Imagine numerous post-it notes with passwords and URLs blowing around an office. Configuration management solves this challenge by creating a "source of truth" with a central location for configuration.

Git is a fantastic platform for managing configuration data. Moving configuration data into a Git repository enables version control and the repository to act as a source of truth. Version control also solves another configuration problem: unexpected breaking changes. Managing

problems: unexpected breaking changes. Managing unexpected changes through the use of code review and version control helps to minimize downtime.

Configuration values will often be added, removed, or modified. Without version control this can cause problems. One team member may tweak a hardware allocation value so that the software runs more efficiently on their personal laptop. When the software is later deployed to a production environment, this new configuration may have a suboptimal effect or may break.

Version control and configuration management solve this problem by adding visibility to configuration modifications. When a change is made to configuration data, the version control system tracks it, which allows team members to review an audit trail of modifications.

Configuration version control enables rollback or "undo" functionality to configuration, which helps avoid unexpected breakage. Version control applied to the configuration can be rapidly reverted to a last known stable state.

How configuration management fits with DevOps, CI/CD and agile

Configuration data has historically been hard to wrangle and can easily become an afterthought. It's not really code so it's not immediately put in version control and it's not first-class data so It isn't stored in a primary database. Traditional and small scale system administration is usually done with a collection of scripts and ad-hoc processes. Configuration data can be overlooked at times, but it is critical to system operation.

The rise of cloud infrastructures has led to the development and adoption of new patterns of infrastructure management. Complex, cloud-based system architectures are managed and deployed through the use of configuration data files. These new cloud platforms allow teams to specify the hardware resources and network connections they need provisioned through human and machine readable data files like YAML. The data files are then read and the infrastructure is provisioned in the cloud. This pattern is called infrastructure as code (IaC).

DevOps configuration management

In the early years of internet application development, hardware resources and systems administration were primarily performed manually. System administrators wrangled configuration data while manually provisioning and managing hardware resources based on configuration data.

DevOps configuration is the evolution and automation of the systems administration role, bringing automation to infrastructure management and deployment.

DevOps configuration also brings system administration responsibility under the umbrella of software engineering. Enterprises today utilize it to empower software engineers to request and provision needed resources on demand. This removes a potential organizational dependency bottleneck of a software development team waiting for resources from a separate system administration team.

CI/CD configuration management

Agile configuration management

Configuration management enables agile teams to clearly triage and prioritize configuration work. Examples of configuration work are chores and tasks like:

- Update the production SSL certificates
- Add a new database endpoint
- Change the password for dev, staging, and production email services.
- Add API keys for a new third-party integration

Once a configuration management platform is in place, teams have visibility into the work required for configuration tasks. Configuration management work can be identified as dependencies for other work and properly addressed as part of agile sprints.

addressed as part of agile sprints.

Configuration management tools

Git

Git is the industry-leading version control system to track code changes. Adding configuration management data alongside code in a Git repository provides a holistic version control view of an entire project. Git is a foundational tool in higher-level configuration management. The following list of other configuration management tools is designed to be stored in a Git repository and leverage Git version control tracking.

Docker

Docker introduced containerization that is an advanced form of configuration management -- like a configuration lockdown. Docker is based on configuration files called Dockerfiles, which contain a list of commands that are evaluated to reconstruct the expected snapshot of operating system state. Docker creates containers from these Dockerfiles that are snapshots of a preconfigured application. Dockerfiles are committed to a Git repository for version tracking and need additional configuration management to deploy them to infrastructure.

Terraform

Terraform is an open source configuration management platform by HasiCorp. Terraform uses IaC to provision and manage clusters, cloud infrastructure, or services. Terraform supports Amazon Web Services (AWS), Microsoft Azure, and other cloud platforms. Each cloud platform has its own representation and interface for common infrastructure components like servers, databases, and queues. Terraform built an abstraction layer of configuration tools for cloud platforms that enable teams to write files that are reproducible definitions of their infrastructure.

Ansible, Salt Stack, Chef, Puppet

Ansible, Salt Stack, and Chef are IT automation frameworks. These frameworks automate many traditional system administrators' processes. Each framework uses a series of configuration data files -- usually YAML or XML -- that are evaluated by an executable.

The configuration data files specify a sequence of actions to take to configure a system. The actions are then run by the executable. The executable differs in language between the systems -- Ansible and Salt Stack are Python based and Chef is Ruby. This workflow is similar to running ad-hoc shell scripts but offers a more structured and refined experience through the respective platforms ecosystems. These tools are what will bring enable the automation needed to achieve CI/CD.

How to implement configuration management

Identification

The first action towards configuration management is information gathering. Configuration data should be aggregated and compiled from different application environments, development, staging, and production for all the components and services in use. Any secret data like passwords and keys should be identified and securely encrypted and stored. At this point configuration data should be organized into data files that can be pointed to as a central source of truth.

Baseline

After configuration data has been aggregated and organized a baseline can be established. A baseline

configuration is a known state of configuration that will successfully operate the dependent software without error. This baseline is usually created by reviewing the configuration of a functioning production environment and committing those configuration settings.

Version Control

Your development project should use a version control system. If not, install Git, initialize a repository for the project, and add the configuration data files to the repository. A word of caution before adding configuration data to a repository: make sure that any secret data like passwords or keys are encrypted with an external key. Secret data accidentally committed to a repository is a huge risk. It needs to be scrubbed from the repositories history or it will be at risk of being exploited.

Auditing

Having configuration data organized and added to a repository enables collaboration and visibility into the system's configuration. The popular pull request workflow that software teams use to review and edit code can then be applied to configuration data files. This helps build out an audit and accounting system. Any changes applied to the configuration must be reviewed and accepted by the team. This adds accountability and visibility into configuration changes.

In conclusion...

Configuration management is a necessary tool for managing complex software systems. Lack of configuration management can cause serious problems with reliability, uptime, and the ability to scale a system. Many current software development tools have configuration management features built in. Bitbucket offers a powerful system for configuration management that is built around Git pull request workflows and CI/CD pipelines.

SHARE THIS ARTICLE

What is Continuous Integration

Software testing for continuous delivery

What Is Continuous Deployment?

Microservices and Microservices Architecture

Bitbucket CI/CD tutorials

Continuous Delivery articles

IAN BUCHANAN

While Ian has broad and deep experience with both Java and .NET, he's best known as a champion of agile methods in large enterprises. He's currently focused on the emerging DevOps culture and the tools for enabling better continuous integration, continuous delivery, and data analysis. During his career, he's successfully managed enterprise software development tools in all phases of their lifecycle. He has driven organization-wide process improvement with results of greater productivity, higher quality, and improved customer satisfaction. He has built multi-national teams that value self-direction and self-organization. When not speaking or coding, you can find Ian indulging his passions in parsers, meta-programming, and domain-specific languages. Follow Ian at @devpartisan.

ARTICLE

Value Stream Mapping

Value stream mapping is an analysis technique that can help optimize your continuous delivery pipeline. Learn how and why this technique is used.

Read this article →

ARTICLE

DevSecOps: Injecting Security into CD Pipelines

You've heard of DevOps, but what is DevSecOps? Hint: it has to do with security. Learn more as we discuss this trend in continuous delivery.

Read this article →

CI/CD Topics

Continuous Delivery
Continuous Integration
Continuous Deployment
Pipelines
Software Testing
Microservices
Tutorials

Sign up for more CI/CD articles and tutorials.

Email