Infrastructure Automation

ARMS
Applied Resilience for Mission Systems

Applied Resilience for Mission Systems

Mark Rabe

Overview

Commercial industry has been rapidly evolving the way it builds, operates, and maintains computer infrastructure to support resilient, scalable systems. This evolution has resulted in a multitude of benefits, including quicker development, deployment, and re-deployment cycles for software. The government has been slow to adopt these new infrastructure paradigms and continues to use largely traditional and manual configuration management and system deployment approaches to manage its servers, virtual machines (VMs), and related IT infrastructure. The issue with these older approaches is that creating, deploying, configuring, maintaining, and recovering systems is an error-prone process (magnified during crisis), that consumes a lot of time and involves a lot of repeated effort across multiple systems.

In order to create a robust infrastructure, leveraging the industry's Infrastructure as Code (IaC)[1,2], an automated infrastructure capability and supporting process, is needed. IaC can be broken down into three categories of tools: Revision Control, Base Image Management, and Infrastructure Deployment. This document will provide a brief overview of these categories and the benefits to using IaC.

Need

There are a number of undesirable practices and approaches to infrastructure that relate to the older, more traditional way of managing systems. Traditionally, DoD systems have primarily been manually deployed and maintained. Although the DoD has been adopting various aspects of automation and efforts like the DevSecOps Initiative (http://dccscr.dsop.io) are focused on bringing some of these practices to the DoD, they are in their early stages. Manual actions are highly error-prone, particularly during incident response. Surprise disruptions (hardware failures, environmental or adversarial attack) can result in long downtimes to rebuild systems. Also, almost all systems have discrepancies between infrastructure documentation and implementation that accumulate over time.

Approach

We apply IaC concepts and processes to automate the management and provisioning of our operational deployments. IaC is the ability to configure your architecture using machine-readable files (code) as opposed to a manual/interactive approach[3].

To give an example of this, we use IaC to deploy a surrogate mission system with security and test monitoring enclaves into AWS. We use Packer to create our base images (CentOS 7 hardened images) that define our common packages, configuration, and users. We use Terraform to create all the AWS infrastructure, including virtual private clouds, routing tables, internet gateways, network access control lists, security groups, subnets, and VM instances (using the base images). We then use Ansible to install software and configure each VM instance based on its role (security monitoring, test monitoring, disruption, analysis, service, etc.). We use the following steps.

Step 1: Develop Infrastructure Plan and Code

Develop an initial plan for how mission capability will be deployed onto the IT resources and give some thought to how the various repositories will be organized before developing the infrastructure automation code. The infrastructure code consists of all the actions needed to deploy the base infrastructure and configure the systems within that infrastructure *in code form* (e.g., networks, VM instances, identities, access control, etc.). The programming language used to write the code will depend on the tools being used to perform the infrastructure automation. We use Terraform to configure and deploy the infrastructure components (e.g., networking) and deploy base VM instances. Terraform interacts with multiple datacenter infrastructures (e.g., VMWare, AWS, Azure, etc.). We use Ansible to configure the instances (e.g., OS, applications, etc.).

[1] B. Johnson, "Introduction to Infrastructure as Code," https://www.networkcomputing.com/networking/introduction-infrastructure-code

[2] M. Zamot, "How to use infrastructure as code," https://opensource.com/article/19/7/infrastructure-code

[3] "Infrastructure as Code," https://en.wikipedia.org/wiki/Infrastructure_as_code

For more information contact:
ARMS@LL.MIT.EDU

LINCOLN LABORATORY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Infrastructure Automation

ARMS
Applied Resilience for Mission Systems

Step 2: Revision Control

While developing the code, it is important to have a strategy to manage that code. A revision control system allows for changes to the infrastructure code to be tracked and managed over time like any other software. These systems allow multiple developers to collaborate on the same code base and revert to previous versions.

Step 3: Base Image Management

Base images (operating system images, container images, etc.) can be created and tailored (e.g., applying security controls or other compliance requirements) to the unique needs of different environments; custom-tailored images are more trustworthy than existing images from the internet. We use Hashicorp Packer to build our base images for VMWare, AWS, and Docker.

Step 4: Infrastructure Deployment

Infrastructure deployment is broken down into two different stages, described below (see Figure 1). Infrastructure code will be modularized for reuse across multiple environments and applications.

Stage 1 (Terraform): deploy base systems and supporting infrastructure components (networking, security, cloud-specific elements) to multiple public and private cloud environments.

Stage 2 (Ansible): configure the systems once deployed, and reuse across multiple environments or other deployments. Additional security controls can be applied to applications on individual systems.

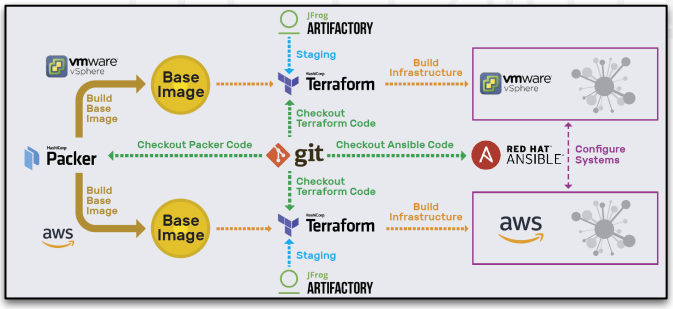

Figure 1. IaC process.

Impact

By prototyping and implementing deployment automation practices, we are aiming to modernize system-building practices in the DoD. This modernization improves the resiliency of the mission systems as it makes them more understandable and manageable and reduces the time to re-configure and re-deploy systems.

Infrastructure Code as Documentation. Having the infrastructure as code allows for the infrastructure to be fully defined in a readable format and effectively serve as the ground-truth documentation of the architecture. It also creates a common view of the infrastructure across all stakeholders (e.g., administrators, accreditors).

Code Review. Revision control allows for any changes to the infrastructure code to be reviewed before being applied.

Code Versioning and Tagging. Revision control systems allow the infrastructure code to be versioned or tagged when there is a stable release, allowing for easy rollback to a known good state. This increases the confidence of infrastructure engineers to make changes to the system.

Base Image Tailoring. Creating base images for environments allows infrastructure engineers to tailor the image to different environments with different levels of security/compliance (e.g., DISA STIGs[4]).

Base Image Provenance. Creating base images from scratch allows infrastructure engineers to have more confidence in their images (no maliciously tampering).

Reusable across Programs/Environments. Codified infrastructure allows for all or parts of that code to be shared with other organizations and programs. It also allows for the same deployment to be deployed to multiple types of environments (development, testing, and production) and across multiple classifications.

Rapid Restoration. Rapid deployment enables restoring the architecture to a known good/clean state or previous version should an issue arise and also supports resiliency processes like Chaos Engineering.

Rapid Adaptation. Rapid deployment also enables the system to be changed/adapted quickly. This allows for patches/changes to be introduced quickly.

Immutable Servers[5]. Server immutability prevents drift from the infrastructure code and what is deployed.

What Should Builders Do

- Codify your IT infrastructure and use Infrastructure as Code tools to manage and deploy your capabilities
- Manage the infrastructure code in revision control

4 "DISA Security Technical Implementation Guide," https://public.cyber.mil/stigs/

5 K. Morris, "ImmutableServer," [Online]. Available: https://martinfowler.com/bliki/ImmutableServer.html

This material is based upon work supported by the United States Air Force under Air Force Contract No. FA8702- 15-D-0001. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Air Force.

LINCOLN LABORATORY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY