5.1 Deployment

Deployment

- Once that the code has been built and tested, we need to deploy it to our servers so that our customers can use the newly developed features!
- There are many competing tools and options in this space, and the one that is right for you and your organization will depend on your needs.
  - Puppet
  - Ansible
  - Salt
  - PalletOps etc.

Deployment

- Focus on installing binary packages and their configuration with a configuration management system on actual servers (client-side code).

Let enterprise application, with a number of different high-level components and

  - A web server
  - An application server
  - A database server

Deployment

- If only single physical server and these few components to worry about that get released once a year or so, we can install the software manually and be done with the task.
- It will be the most cost-effective way of dealing with the situation, even though manual work is boring and error prone.
- It is more likely that a large organization has hundreds of servers and applications and that they are all deployed differently, with different requirements.

Deployment

- Managing all the complexity that the real world displays is hard, so it starts to make sense that there are a lot of different solutions that do basically the same thing in different ways.
- Fundamental unit that executes our code can be
  - a physical server
  - a virtual machine
  - some form of container technology
  - or a combination of these
- we have several challenges to deal with.

Configuring the base OS

- Some application stacks, such as Java, Python, or Ruby, make these operating system requirements less apparent, because these technologies go to a great length to offer cross-platform functionality.
- At other times, the operating system requirements are apparent to a greater degree, such as when you work with low-level mixed hardware and software integrations, which is common in the telecom industry.

Configuring the base OS

- Some systems work with a bare metal (or bare virtual machine) approach, where they install the desired operating system from scratch and then install all the base dependencies that the organization needs for their servers.
- Such systems include, for example, Red Hat Satellite and Cobbler, which works in a similar way but is more lightweight.
- Cobbler allows you to boot a physical or virtual machine over the network using dhcpd.
- The DHCP server can then allow you to provide a netboot-compliant image. When the netboot image is started, it contacts Cobbler to retrieve the packages that will be installed in order to create the new operating system. Which packages are installed can be decided on the server from the target machine's network MAC address for instance.

Configuring the base OS

- Another method that is very popular today is to provide base operating system images that can be reused between machines.
- Cloud systems such as AWS, Azure, or OpenStack work this way. When you ask the cloud system for a new virtual machine, it is created using an existing image as a base.
- Container systems such as Docker also work in a similar way, where you declare your base container image and then describe the changes you want to formulate for your own image.

Describing clusters

- If your organization only has a single machine with a single application, then you might not need to describe how a cluster deployment of your application would look like.
- Unfortunately (or fortunately, depending on your outlook), the reality is normally that your applications are spread out over a set of machines, virtual or physical.
- Puppet has an extensive system that allows machines to have different roles that in turn imply a set of packages and configurations.
- Ansible and Salt have these systems as well.

Describing clusters

- The container-based Docker system has an emerging infrastructure for describing sets of containers connected together and Docker hosts that can accept and deploy such cluster descriptors.
- Cloud systems such as AWS also have methods and descriptors for cluster deployments.
- Cluster descriptors are normally also used to describe the application layer.

Delivering packages to a system

- Much of an application can be installed as packages, which are installed unmodified on the target system by the configuration management system.
- Package systems such as RPM and deb have useful features, such as verifying that the files provided by a package are not tampered with on a target system, by providing checksums for all files in the package.
- This is useful for security reasons as well as debugging purposes.

Delivering packages to a system

- Package delivery is usually done with operating system facilities such as yum package channels on Red Hat based systems, but sometimes, the configuration management system can also deliver packages and files with its own facilities.
- The most common and flexible system to configure applications relies on text-based configuration files.
- There are several other methods, such as using an application that provides an API to handle configuration (such as a command-line interface) or sometimes handling configuration via database settings.

Delivering packages to a system

- There are many ways to manage text-based configurations.
- You can manage them in source code handling systems such as Git.
- There's a host of tools that can ease the debugging of broken configuration, such as diff.
- you can also edit configurations directly on the servers using a remote text editor such as Emacs or Vi.

Delivering packages to a system

- Handling configurations via databases is much less flexible.
- This is arguably an anti-pattern that usually occurs in organizations where the psychological rift between developer teams and operations teams are too wide, which is something we aim to solve with DevOps.
- Handling configurations in databases makes the application stack harder to get running. You need a working database to even start the application.

Delivering packages to a system

- Managing configuration settings via imperative command-line APIs is also a dubious practice for similar reasons but can sometimes be helpful, especially if the API is used to manage an underlying text-based configuration.
- Many of the configuration management systems, such as Puppet, depend on being able to manage declarative configurations.
- If we manage the configuration state via other mechanisms, such as command-line imperative API, Puppet loses many of its benefits.

Delivering packages to a system

- Even managing text-based configuration files can be a hassle. There are many ways for applications to invent their own configuration file formats, but there are a set of base file formats that are popular.
- Such file formats include XML, YML, JSON, and INI.
- Usually, configuration files are not static, because if they were, you could just deploy them with your package system like any piece of binary artifact.
- Normally, the application configuration files need to be based on some kind of template file that is later instantiated into a form suitable for the machine where the application is being deployed.
- An example might be an application's database connector descriptor.

Delivering packages to a system

- If you are deploying your application to a test environment, you want the connector descriptor to point to a test database server.
- Vice versa, if you are deploying to a production server, you want your connector to point to a production database server.
- As an aside, some organizations try to handle this situation by managing their DNS servers, such that an example database DNS alias database.yourorg.com resolves to different servers depending on the environment.

Delivering packages to a system

- Being able to use different DNS resolvers depending on the environment is a useful strategy.
- It can be difficult for a developer, however, to use the equivalent mechanism on his or her own development machine.
- Running a private DNS server on a development machine can be difficult, and managing a local host file can also prove cumbersome.
- In these cases, it might be simpler to make the application have configurable settings for database hosts and other backend systems at the application level.