Server Software

Scalable Software @Duke
Infrastructure

• Mostly VMware ESXi on Bare Metal
• 2.5k Linux VMs
• 800 Windows VMs
• 500 Docker containers
OS Evolution @ Duke

- 15 yrs ago - Hand crafted Unix (AIX, Solaris, Linux), minimal Windows
- 10 yrs ago - AIX/Solaris usage slowed, Linux took over and became the most used OS in the transition to VMs
- 5 yrs ago - Majority of software started running directly on Linux VMs
- Now - The rise of Docker abstracted software to containers
OS and Software Management
@Duke

- Puppet for OS
- Ansible for misc. software deployments
- Capistrano for Ruby Deployments
- Gitlab + (Jenkins or Gitlab-CI) for Docker containers
- Stevedore
Puppet

• Declare the state of a server
  – User Logins
  – Firewall ports
  – LDAP/AD Configuration
  – Logs
  – etc
Ansible

• Configure multiple hosts for clustering
• Perform multistep tasks for software installations
• Example:
  – Install new clustered software
  – Initiate cluster management through a rest call
  – Configure cluster with newly bootstrapped credentials
Capistrano

• Ruby-on-rails focused deployments
• Easy to deploy from your laptop to a server
• Caveats
  – Software on server must match your laptop
  – Permission issues
Gitlab(-ci)/Jenkins

- Gitlab-CI is built in to current gitlab instances
  - Allows for seamless continuous integration
  - Requires runners, which can be set up by users
- Jenkins
  - More complex and mature CI
  - Harder to stand up on your own
OIT Docker Workflow

- All development done locally on developer laptops using docker-compose
- Code is checked in to gitlab
- Jenkins picks up the code
  - Runs security checks
  - Runs accessibility checks
  - Builds and pushes container to a docker registry
- Stevedore pulls the newly built container to a set of servers and runs it
Other Management Software

- Kubernetes
- OpenShift
- Chef
- Saltstack
Typical Software Stack

• Storage
  – Local block devices
  – Network Storage
  – Object Store
• Frontend
  – Apache/Nginx
• Application Stack
  – Rails, Django, Go framework
Typical Software Stack (cont.)

- **Database**
  - MariaDB, PostgreSQL, Oracle, MSSQL
- **Authentication**
  - Shibboleth, OIDC
Securing your App

- Start with a secure app
- TLS for all connections
  - https, ssh, scp, ftps is good
  - http, ftp, telnet is bad
- Strong Passwords
  - Human generated passwords are all bad
- Multifactor Authentication
Scaling your Application

• Eliminate single points of failure by ensuring multiple instances of your stack stay up
  – Multihost database
  – Multihost storage
  – Load Balanced frontend, etc

• For further redundancy, deploy in multiple datacenters

• For further further redundancy, go multiple datacenters + cloud providers
What else?

• Backups
• Historical monitoring
  – Performance of your application
  – CPU/Memory
  – Traffic
• Alert monitoring
  – Oops, something crashed
Demo Deployment

- Gitlab
- Jenkins
- Stevedore
- APP!
Links

- https://gitlab.oit.duke.edu/
- https://vcm.duke.edu/
- http://dukecolab.slack.com/
- https://gitlab.oit.duke.edu/devil-ops
- Drew.Stinnett@duke.edu (That’s me!)