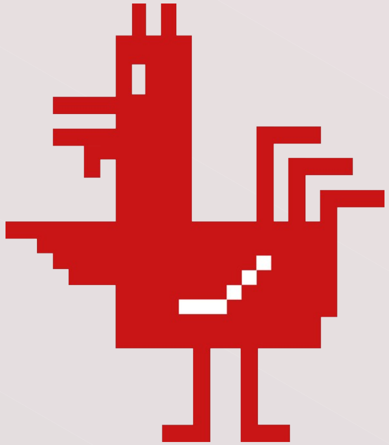
Puppeteers webinar: Infrastructure as Code

Puppeteers

Webinar content

- Traditional way of building infrastructure
- Version control
- Advantages of IaC
- When to use IaC
- Where to start
- Configuration management based on desired state
- Push and pull models
- Tools: Terraform, Puppet, Ansible and Puppet Bolt, Docker
- Quality assurance



Puppeteers

Who are we?

- Puppeteers Oy

  Petri Lammi

  Samuli Seppänen

Puppeteers

Traditional way of building infrastructure - 1

- Graphical tools: Windows, web applications
- Command prompt: UNIX-compatible systems
- Every change needs to be performed separately: change that need to repeated are usually left undone

Puppeteers

Traditional way of building infrastructure - 2

- Inconsistency: everyone does things their own way
- Lots of documentation needed (and it lags behind)
- No visibility to the state of infrastructure
- When making changes need to check the current status first

Traditional way of building infrastructure - 3

- Easy to make mistakes
- Errors spotted usually long after they were made
- Lots of tracking and fixing of issues

Version control

- Essential feature when building infrastructure with code
- Enables several persons to co-operate efficiently in infrastructure development
- Enables change management and quality assurance

Advantages of infrastructure built with code

- Visibility to current status of the systems
- Uniformity
- Less errors
- Errors spotted soon

Advantages of infrastructure built with code - 1

- Repeating changes are made faster
- Spotting anomalies is easier
- Changes are easily reverted

Advantages of infrastructure built with code - 2

- Amount of routine maintenance is reduced a lot
- Inventory
- Control over the whole system lifecycle, if needed

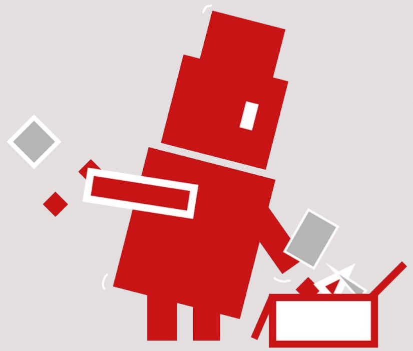
When to use IaC?

- ”Always” but especially in environments that have a lot of repeating configurations

Puppeteers

Where to start?

- Repeating configurations: monitoring, backup, default settings
- Automate other things later on



Puppeteers

Configuration management based on desired state - 1

- Define desired state with code
- Object are forced from current to desired state
- No running of commands or scripts

Configuration management based on desired state - 2

- Desired state consists of atomic parts that are combined and linked together
- Servers/workstations: file, package, service etc.
- Cloud: server instances, routers, domains, etc.

Push and pull models

- Push: state is updated from outside of the managed objects from time to time
- Pull: managed objects auto-update themselves periodically (agents)

Terraform: managing cloud resources

- AWS, Azure, GCP, Digital Ocean, Rackspace, Hetzner, etc.
- Cloud resource management and integration with one tool (push)

Puppet: servers and workstations

- Linux, Windows, MacOS X, *BSD, etc.
- Packages, files, services, DSC resources, etc.
- Either pull (puppetserver + agents) or push (puppet bolt)

Ansible and Puppet Bolt: orchestration

- For orchestration and also state management
- In cases in which the order of changes is as important as the modeled desired state
- Push

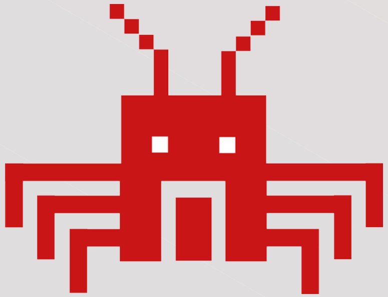
Docker

- Software packaging format, usually built with shell commands in the Dockerfile
- Used as ”lightweight virtual machines” but the term is misleading
- Different components of an application (webserver, database) should be seperate containers
- Container orchestration solutions (Kubernetes etc)

Puppeteers

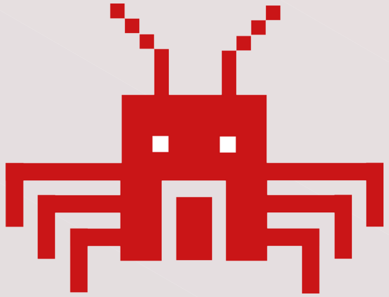
Quality assurance - 1

- Automatic validation tools (validation, linting)
- Unit testing (e.g. rspec)
- Acceptance tests (e.g. Beaker, Litmus, ServerSpec)



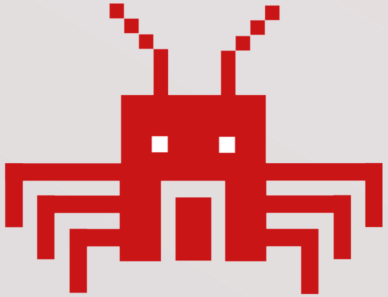
Quality assurance - 2

- Testing change in production in no-operation mode
- Testing with a single production node
- Deployment to production
- Monitoring of the effect of the changes



Quality assurance - 3

- Development environments (e.g. Vagrant or Docker)
- Dedicated test environments (e.g.Terraform)
- Forced code reviews



Puppeteers

Thank you!

- Webinar series continues
- https://www.puppeteers.net

Puppeteers

Petri Lammi

Samuli Seppänen