DSPACE & PUPPET

CLUSTERING UND CUSTOMISATION ZENTRAL VERWALTEN
TABLE OF CONTENTS

1. What we want
2. What DSpace needs
3. How we get what we want
4. Example
5. What we need to know
WHAT WE WANT
• Automatic, customizable and secure all-in-one or cluster configuration
• Horizontal scaling, improved availability and improved maintainability via Clustering
• Integration into central authentication, monitoring and backup/archive services
WHAT DSPACE NEEDS
Software stack

- Apache web server
- Tomcat application server
- PostgreSQL database server
- Solr index server
- DSpace web application
- File system
Installation of source code

- Build: Maven
- Install: Ant
Customisations in

- Configuration files
- Source code
HOW WE GET WHAT WE WANT
Software stack

- GitLab: Source Code Management & CI & CD
- Puppet: Configuration Management & Orchestration
- Custom Puppet module for DSpace
Custom Puppet module for DSpace

- Handles installation & configuration of all DSpace components
- Handles firewall, orchestration and communication in cluster setting
GitLab + Puppet:
Characteristics of our Clustering

- The components portal, application, database and index are installed and configured on designated nodes
- All DSpace-specific server components listen to local host
- Only Web server listens on the public interface
Characteristics of our Clustering

- In cluster mode all components are transparently connected via HAProxy
- Secure firewall configuration
Configuration and customization via parametrized Puppet classes

- Configuration of basic System/Java/Tomcat environment variables (*dspace::environment*)
- Configuration of basic DSpace-related settings via (*dspace::configuration*)
- Configuration of DSpace-component related things (*dspace::portal, dspace::application, dspace::database, dspace::index*)
Configuration and customization via Puppet module, custom Git repositories and Maven overlay

- Git repository with slightly modified basic DSpace source code
- Git repository with different branches of DSpace Maven overlay source code
- Embedded-Ruby templates for `local.cfg` and several other configuration files
A DSpace Application node in Puppet

class { 'dspace::configuration':
    dspace_name => 'My DSpace Repository',
    dspace_base_url => 'https://my-dspace-repository.org',
}

class { 'dspace::application::source':
}

class { 'dspace::application::server':
}

class { 'dspace::application::node':
    cluster => 'my-dspace-cluster',
}
Puppet Agent on a DSpace Application node

\textit{(dspace::application::source)}

- Gets configuration from Puppet Master
- Clones/pulls the basic DSpace Git repository
- Clones/pulls the DSpace maven overlay Git repository
Puppet Agent on a DSpace Application node  
(*dspace::application::server*)

- Fills and places configuration template files
- Runs Maven and Ant
- Restarts Tomcat
Puppet Agent on a DSpace Application node

(dsparse::application::node)

- Sets a HAProxy frontend for communication with frontend web server (dspace::portal::server)
- Sets a HAProxy backend for the database (dspace::database::server)
- Sets a HAProxy backend for the index (dspace::index::server)
Puppet Agent on a DSpace Application node 
(*dspace::application::node*)

- Collects firewall *allow* rule from all web servers 
  (*dspace::portal::server*) in cluster
- Exports a HAPerxy backend to all web servers 
  (*dspace::portal::server*) in cluster
- Restarts HAPerxy
WHAT WE NEED TO KNOW
Shared resources in a cluster setting

- Database and index
- AssetStore directory (we use CephFS)
- Some other directories (e.g. exports, we use CephFS)
- Session data? Could be shared, but currently we just pin the client on a worker
There will be no conflicts/collisions, right?

- DSpace ensures that the asset creation and update process is consistently serialized via database
- But there are some other things to keep in mind
Avoiding conflicts

- When doing a DSpace upgrade, only one application machine should be running.
- Index and maintenance jobs should run only on one node in the cluster at a time.
THANK YOU!
QUESTIONS