Config Management Tool Comparison

Dave Hill
Department of Computing, TU Dublin, Tallaght, Ireland
X00151295@myTUDublin.ie

Introduction

Optimising the time any resources are alive is a key component of an architecture strategy. This research project aimed to test several methods of bootstrapping resources for use and document the findings in order to make educated decisions regarding software deployments and overall architecture design. The tools under test were Ansible, Chef, Puppet and Salt and with this work, a test framework is also made available to allow for future work to build off these findings. The tests done were to create 10 users, 50 files, clone two git repositories, install JDK 8.0 and install and start Apache webserver.

Cutting Chef

As part of the initial assessment of the tools, efforts were made to ensure that the steps needed to set up the master node, retrieve the configuration files needed to run the tests and registering new slaves could all be automated so to be integrated with the test framework delivered as part of this research. For the other tools, this required some effort and research but the level of complexity needed to automate the set up of Chef and integrate with the framework resulted in it's eventual exclusion from the automated testing. This is unfortunate as the manual execution of the tests showed that Chef was certainly one of the most performant options available.

Framework Implementation Details

The test framework uses a combination of Terraform to spin up master and slave nodes, a custom built lambda function used to minimise the time needed from provisioning of the slave nodes to having them register on the master, and a selection of bash scripts which would run on the master or slave nodes, or both depending on the tool in use. The framework is intended to be platform agnostic and can be run against AWS or Azure to spin up slave nodes.
The Registration API was built as a mechanism to avoid delays between polling periods from the master node to find slave nodes and sent the IP address, tool used, test to execute and platform the slave ran on to the master node to drive the logic of the framework
Part of the functionality of the framework and why a custom one was built was to allow tracking of various points of provisioning and execution using timestamps and this, combined with the times output by each of the configuration management tools were all logged to an S3 bucket where the logs for each node were deposited to allow for analysis.

Tool Specific Findings

1. Ansible:

- Extremely lightweight
- Tests easy to write
- Simplest set up of all
- Usability prioritised over Performance

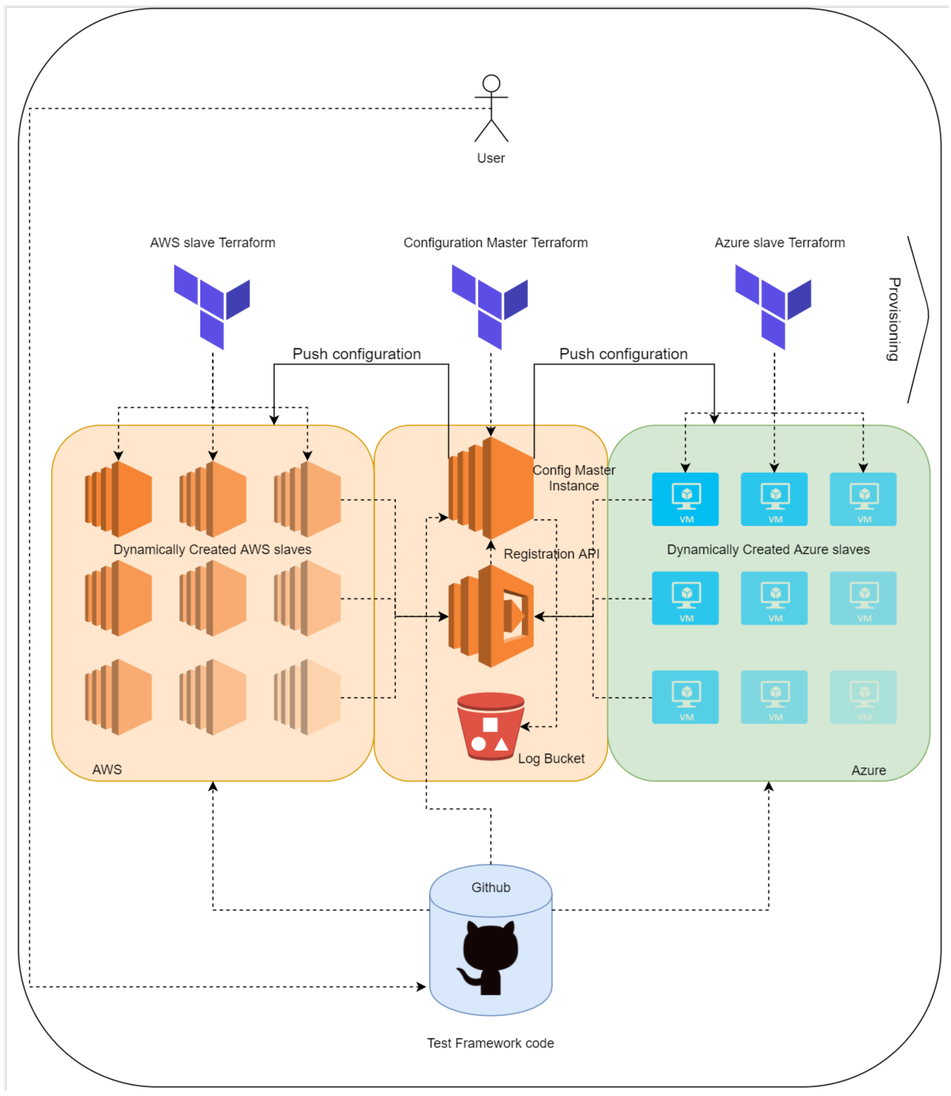
2. Puppet:

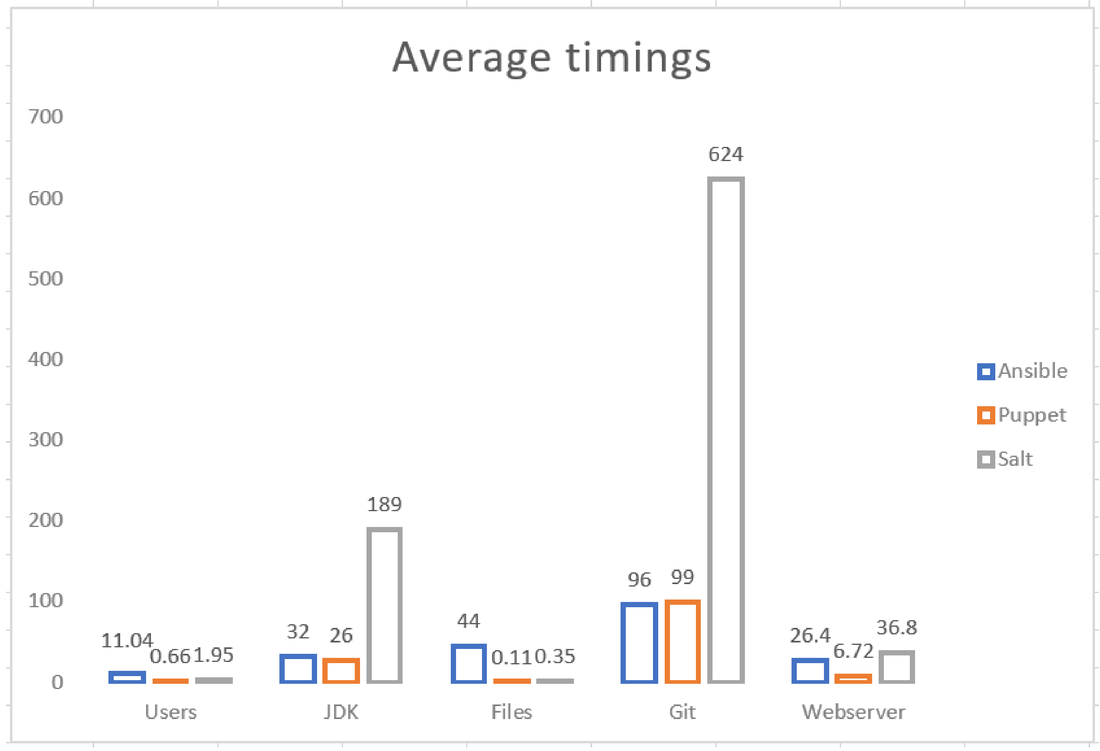
- Fastest tool overall
- Most complex integration
- Heavy enterprise focus in docs
- Tests sourced from community

3. Salt:

- Very easy to set up
- Huge variance in run times
- Tests trickier to write
- Concurrency appears to induce variance

Framework overview and Results*





*Time Measured in Seconds

Conclusions and Future Work

At a high level, this research can say that Puppet is the most consistently high performing tool for the criteria measured as part of this study. Ansible was consistent and capable of competing in some tests but was significantly slower in others. It's set up and maintenance time make it attractive to those with less resources for investment. Salt was capable of incredible performance but as the number of nodes to configure grew, it became wildly inconsistent with some massive differences as seen in the results. When testing cloud platforms, very little variance appeared between AWS and Azure, but AWS was consistently faster but barring the JDK test where Azure was inexplicably much slower, the difference was minor. Future work would include branching into more cloud providers, more configuration tools, more tests and enhancements to the framework to make log analysis more efficient