Spread

The testing tool behind snaps

November 2022

CANONICAL ubuntu

Testing must adapt and scale

Automated testing is crucial

Some important aspects :

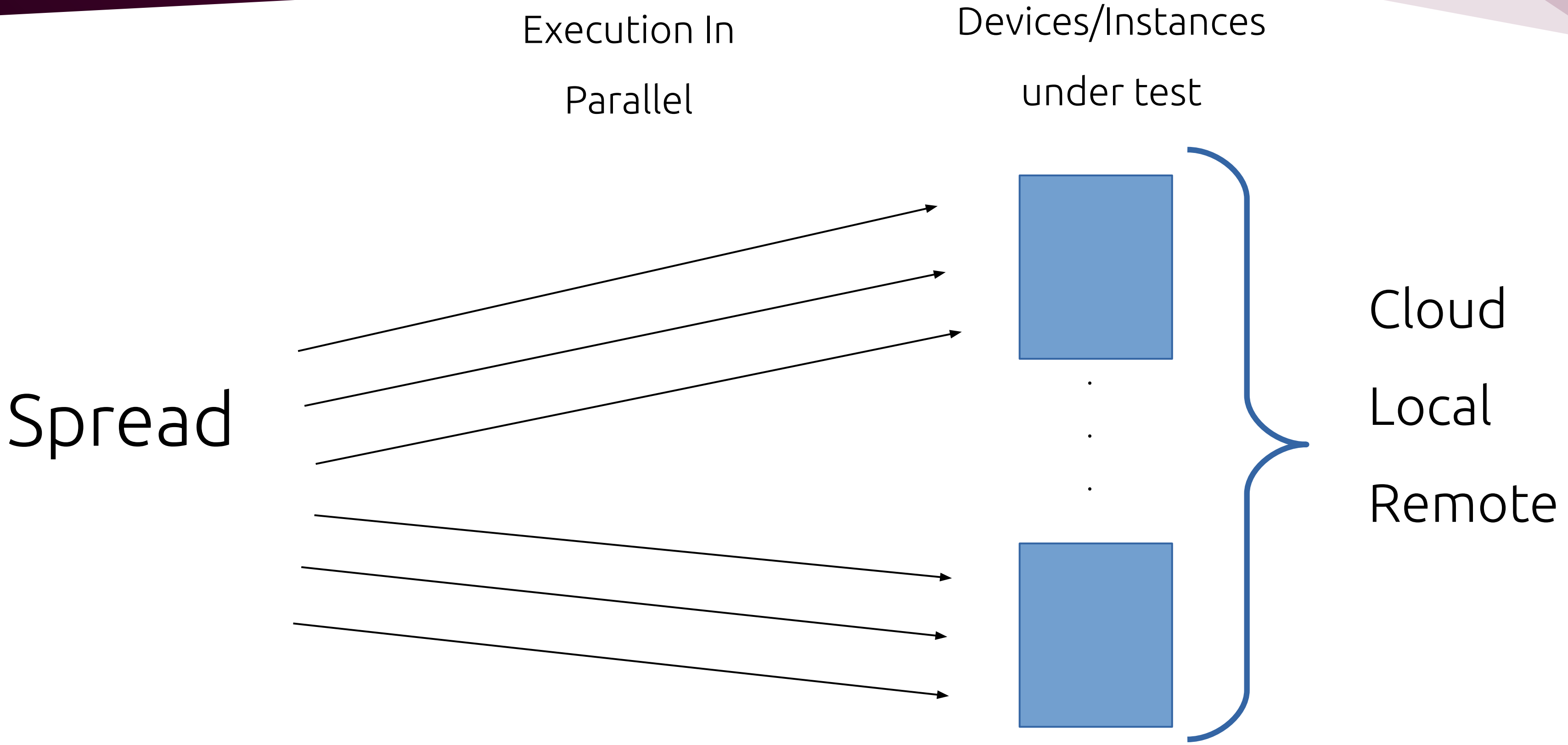
- Fast
- Reliable
- Easy to write tests and understand the output
- Debug and reproduce errors
- Allow to test in various Operating Systems

What is Spread?

Spread is a tool to run tasks

- Allow parallel execution
- Support Containers, Cloud providers, etc
- Support different Operating Systems
- Not specific to testing
- Allow debugging
- Easy to integrate in CI

How it works?



Execution In Parallel

Devices/Instances under test

Spread

Cloud

Local

Remote

What

Where

How

How

Spread provides a CLI and a configuration file to:

- Specify backends, systems, suites and tasks to run in a single command
  - Repeat executions
  - Debug when a task fails
  - Shell into the target machines
- Configure the project in a single configuration file (spread.yaml)
  - Configure the number of worker (runners in parallel)
  - Specify timeouts
  - Setup how to prepare and restore project, suites and tasks
- ...

Cli examples

1) spread lxd:ubuntu-18.04-64:tests/main/abort
2) spread lxd:ubuntu-18.04-64:tests/main/
3) spread lxd:ubuntu-18.04-64:
4) spread lxd:
5) spread lxd:ubuntu-18.04-64:tests/main/ lxd:ubuntu-20.04-64:tests/main/abort
6) spread -debug google:ubuntu-core-18-64:tests/main/abort
7) spread -shell google:ubuntu-core-18-64:tests/main/abort
8) spread -repeat 10 google:ubuntu-core-18-64:tests/main/abort
9) spread -list google:ubuntu-core-18-64:tests/main/

Config example (spread.yaml)

```
project: snapd

environment:
    GOHOME: /home/gopath
    GOPATH: $GOHOME

backends:
    qemu:
        systems:
          - ubuntu-20.04-64:
               username: ubuntu
               password: ubuntu
    ...

path: /home/gopath
exclude:
    - .git

debug-each: |
    dmesg

kill-timeout: 10m

prepare: |
    echo “Preparing the project”
```

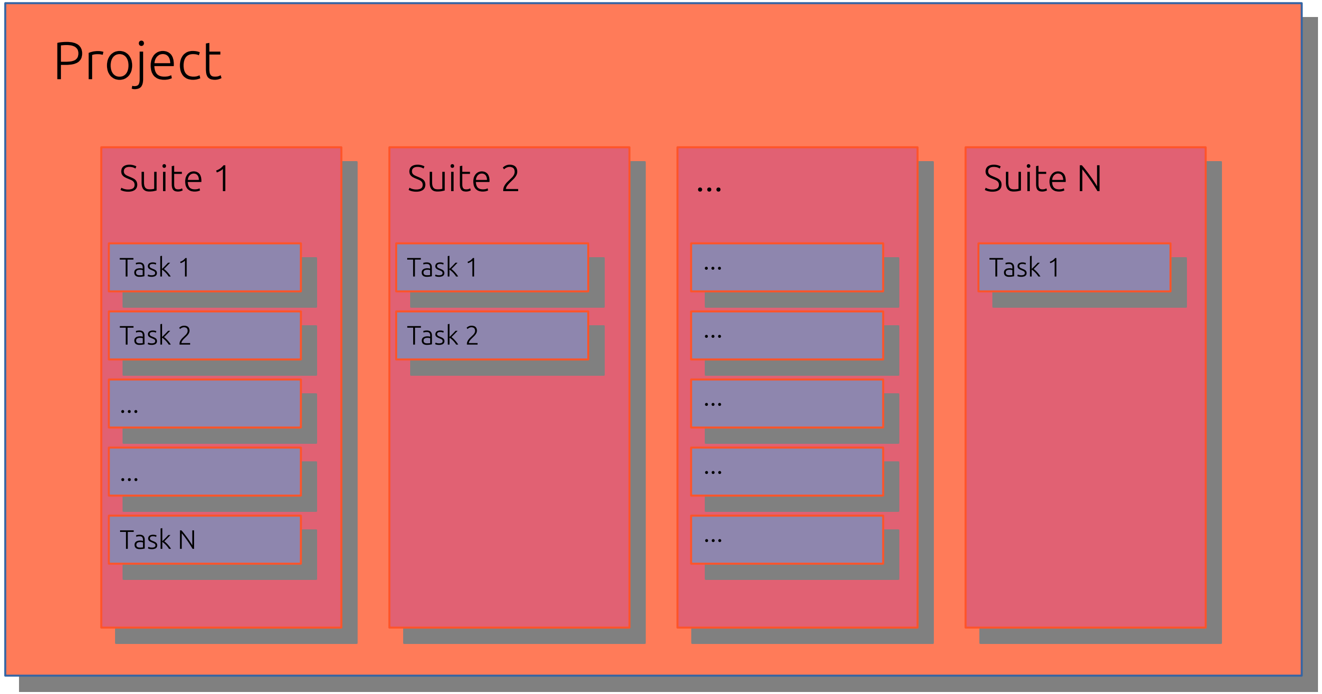
```
restore: |
    echo “Restoring the project”

suites:
    tests/main/:
      summary: This is the main suite
      backends: [google, qemu]
      systems: [ubuntu-*-64, debian-*]
      environment:
          SNAP_NAME: test-snapd-tools
          SNAP_ID: 23554
      prepare: |
          echo “Preparing the main suite”
      prepare-each: |
          echo “Preparing the task”
      restore-each: |
          echo “Restoring the task”
      restore: |
          echo “Restoring the main suite”
```

What

The projects is compose by test suites

Each suite contains 1 or more tasks

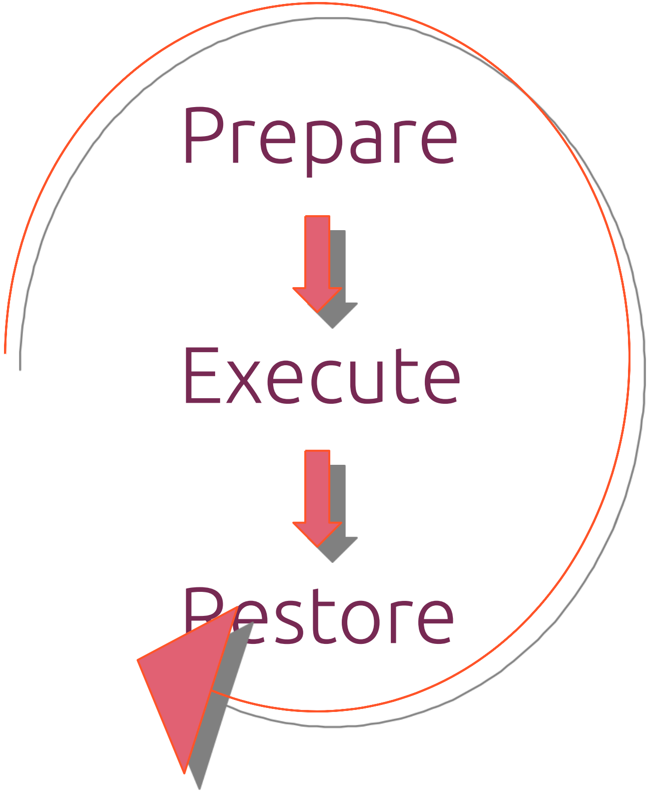
The tasks are executed in the target systems



Execution flow

The project and the suites are prepared and restored

The tasks are also executed



```
Prepare project
 -> Prepare Suite1
 ->-> Prepare Task1
 ->-> Execute Task1
 ->-> Restore Task1
 ->-> Prepare Task2
 ->-> Execute Task2
 ->-> Restore Task2
 -> Restore Suite1
 -> Prepare Suite2
 ->-> Prepare Task3
 ->-> Execute Task3
 ->-> Restore Task3
 -> Restore Suite2
Restore project
```

Data driven and Environment

Pretty much everything in Spread can be customized with environment variables.

The values defined for those variables are evaluated at the remote system

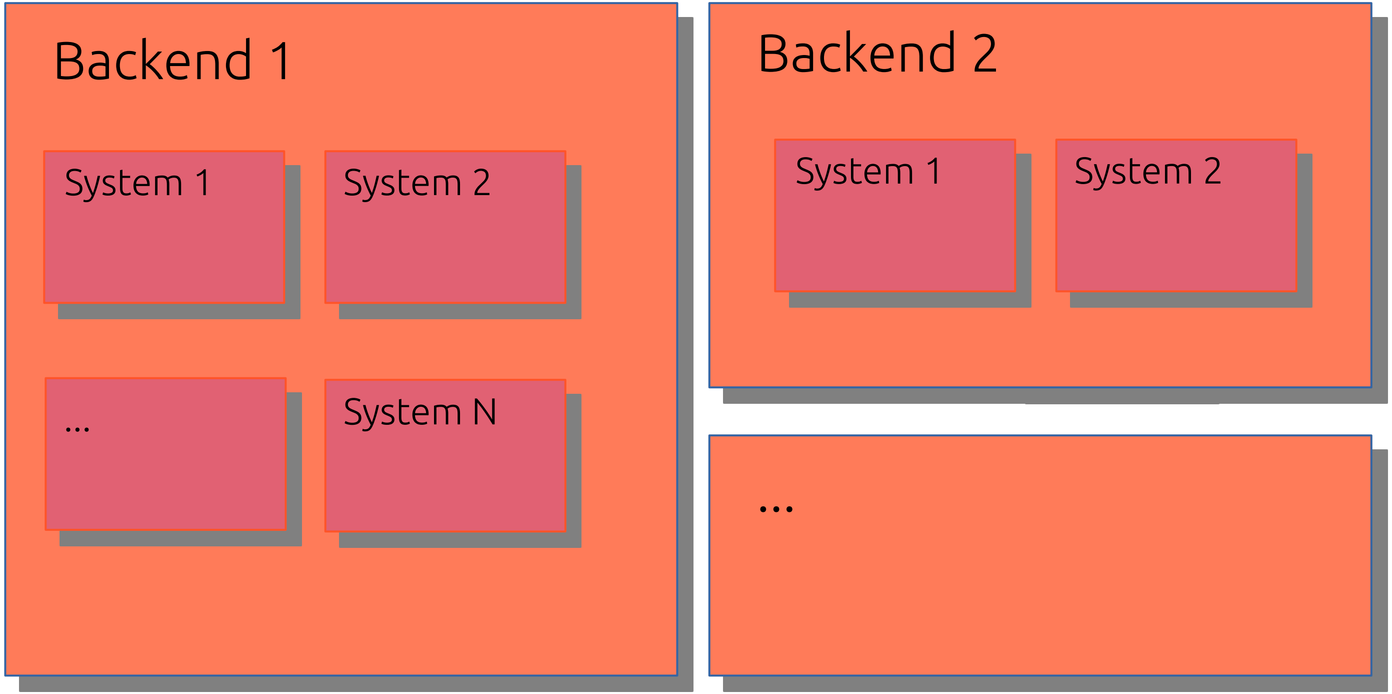
The variants is how spread allow to do data driven testing.

Each variant key produces a single job execution.

Task example

```
summary: Run snap sign to sign a model assertion

systems: [-ubuntu-core-*, -ubuntu-*-ppc64el, -fedora-*, -opensuse-*]

environment:
  VARIANT/stdin: stdin
  VARIANT/file: file

prepare: |
    . "$TESTSLIB"/mkpinentry.sh
    . "$TESTSLIB"/random.sh
    kill_gpg_agent

debug: |
    #shellcheck source=tests/lib/random.sh
    . "$TESTSLIB"/random.sh
    debug_random || true

execute: |
    echo "Creating a new key without a password"
    expect -f create-key.exp

    echo "Ensure we have the new key"
    snap keys|MATCH default
```

Where



Backend 1
System 1
System 2
...
System N
Backend 2
System 1
System 2
...

Available Backends

Spread provides a set of backends to use:

- LXD depends on the LXD container hypervisor available on Ubuntu 16.04 or later, and allows you to run tasks using the local system alone
- QEMU depends on the QEMU emulator available from various sources and allows you to run tasks using the local system alone even if those tasks depend on low-level features
- Ad-hoc (devices) allows scripting the procedure for allocating and deallocating systems directly in the body of the backend
- Google iis easy to setup and use, and allows distributing your tasks to remote infrastructure in Google Compute Engine (GCE)
- Linode is very simple to setup and use as well, and allows distributing your tasks over into remote infrastructure runing in Linode's data centers.
- In progress – AWS and Openstack

Backend/Systems examples

```
qemu:
    systems:
      - ubuntu-16.04-64:
         username: ubuntu
         password: ubuntu
      - ubuntu-18.04-64:
         username: ubuntu
         password: ubuntu
      - ubuntu-18.04-32:
         username: ubuntu
         password: ubuntu
      - ubuntu-20.04-64:
         username: ubuntu
         password: ubuntu
      - ubuntu-22.04-64:
         username: ubuntu
         password: ubuntu
      - centos-7-64:
         username: centos
         password: centos
      - amazon-linux-2-64:
         username: ec2-user
         password: ec2-user
      - opensuse-tumbleweed-64:
         username: opensuse
         password: opensuse
```

```
google:
  key: '$(HOST: echo "$SPREAD_GOOGLE_KEY")'
  location: snapd-spread/us-east1-b
  halt-timeout: 2h
  systems:
    - ubuntu-18.04-64:
       storage: 12G
       workers: 8
    - ubuntu-20.04-64:
       storage: 12G
       workers: 8
    - ubuntu-core-16-64:
       image: ubuntu-16.04-64
       workers: 6
    - ubuntu-core-18-64:
       image: ubuntu-18.04-64
       workers: 6
    - ubuntu-core-20-64:
       image: ubuntu-20.04-64
       workers: 6
       storage: 20G
    - ubuntu-core-22-64:
       image: ubuntu-22.04-64
       workers: 6
       storage: 20G
```

Demo

Our experience in Snapd

- Used as safety net on CI
- Used for development
- Used to validate on devices
- Used to run tests on nested machines

Spread integrated in CI

- Full run on every PR change
- Full run when master changes
- Run nightly tests
- Run cron jobs (other tasks)
- Validate core snaps
- Update Images used for testing
- ...

Spread integrated in CI

- Thanks