FUJITSU

shaping tomorrow with you

OCI Runtime Tools for Container Standardization

Ma Shimiao
<mashimiao.fnst@cn.fujitsu.com>

Agenda

- Background
- OCI Introduction
- Runtime Tools
- Our Contribution
- Future Plans
- Q&A

Background

- Container-based solutions grow rapidly
  - Almost all major IT vendors and cloud providers supply
  - More and more people try to use
- There is a large ecosystem for container
  - Infrastructure vendor
  - Container runtime & orchestration
- Many container runtime technologies
  - Docker
  - Rocket/rkt
  - LXD
  - Hyper
  - …

Ecosystem and Containers

FUJITSU

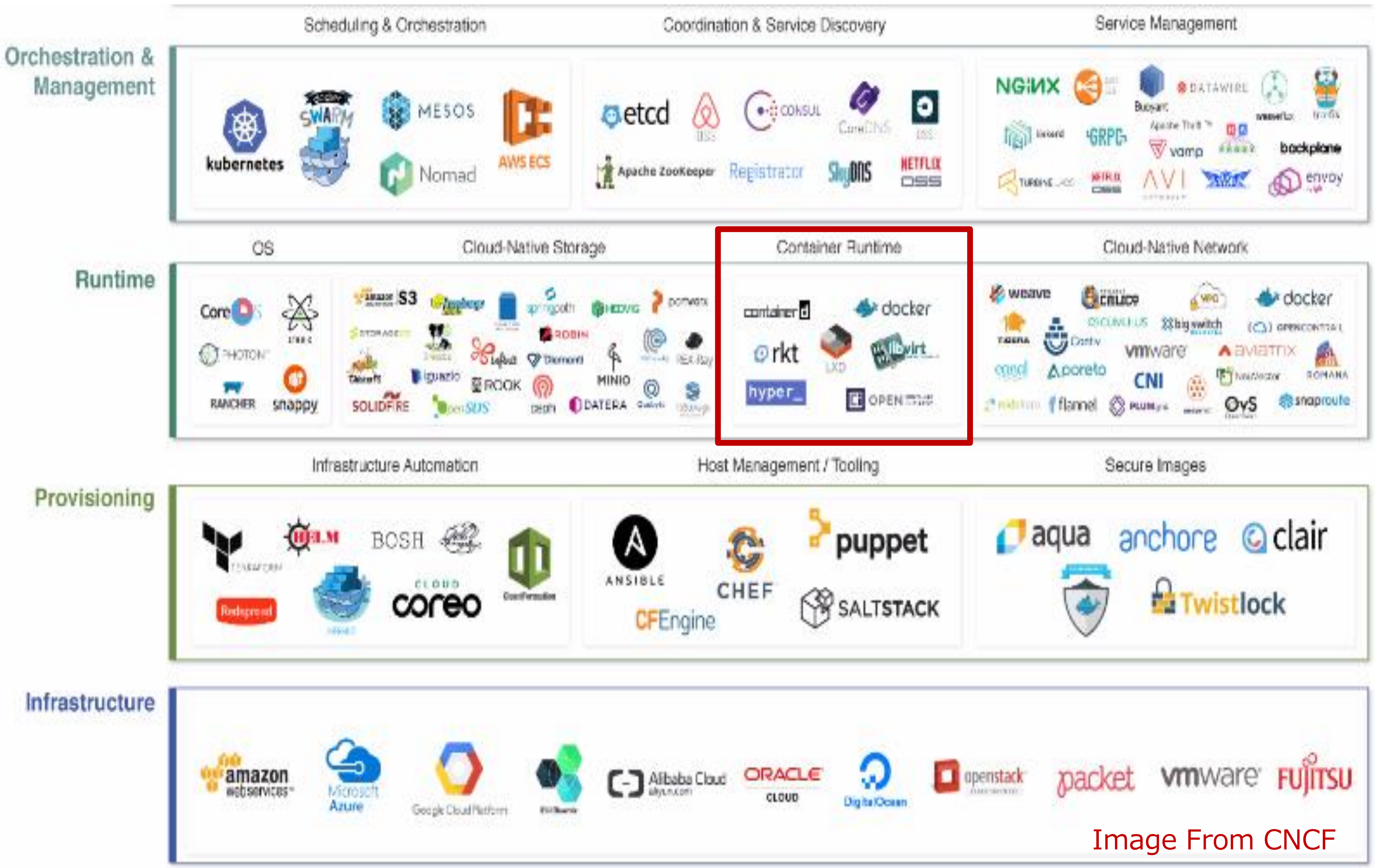


Image From CNCF

Before A Standard

- No open industry standards exist
  - Almost everyone has their own specs
- So, container technology seems to be fragmented
- Users hard to choose the best tools to build the best applications
  - No standards to evaluate
  - Not sure how to evaluate
- Users locked into a technology vendor in the long run
  - Hard to fit difference
  - High cost to transfer applications
- …

Container Standardization

- Make open industry standards for container
  - Unambiguous development direction
  - Portability issue
  - Promote development of container technology
- Help users to choose container-based solutions
  - Users can be guided by choosing the best tools to build the best applications they can
  - Users will not be locked into any technology vendor for the long run
  - Get high quality services

Open Container Initiative

- What is OCI
  - Open Container Initiative, launched on June 22nd 2015
  - a lightweight, open governance structure (project), formed under the auspices of the Linux Foundation
  - 47 members, almost all major of IT vendors and cloud providers
- Mission of the OCI
  - promote and promulgate a set of common, minimal, open standards and specifications around container technology
- Duties of OCI
  - Creating a formal specification for container image formats and runtime
  - Accepting, maintaining and advancing the projects associated with these standards
  - Harmonizing the above-referenced standard with other proposed standards

Projects in GitHub

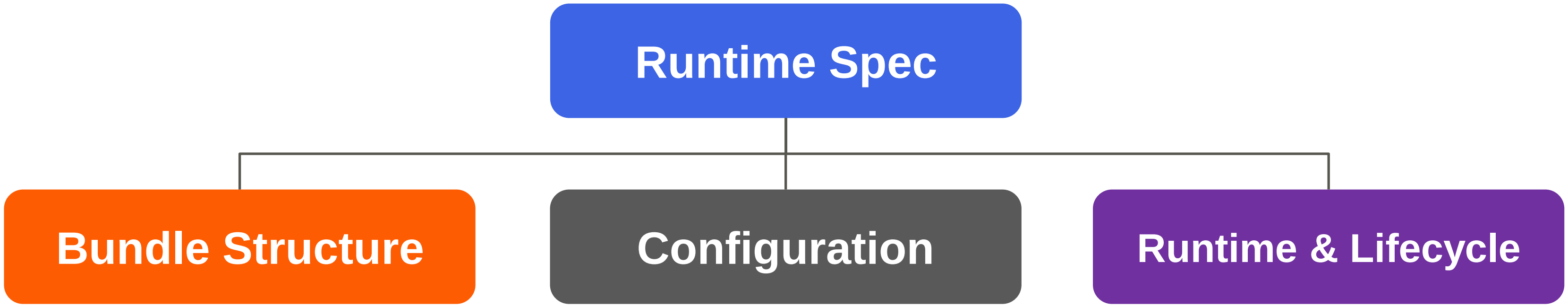
- runtime-spec
  - specifications for standards on Operating System process and application containers
  - http://github.com/opencontainers/runtime-spec
- runtime-tools
  - a collection of tools for working with the OCI runtime specification.
  - http://github.com/opencontainers/runtime-tools
- image-spec
  - creates and maintains the software shipping container image format spec
  - http://github.com/opencontainers/image-spec
- image-tools
  - a collection of tools for working with the OCI image specification.
  - http://github.com/opencontainers/image-tools

Projects in GitHub

- runc
  - a CLI tool for spawning and running containers according to the OCI specification
  - http://github.com/opencontainers/runc
- go-digest
  - common digest package used across container ecosystem
  - http://github.com/opencontainers/go-digest
- go-selinux
  - common SELinux package used across container ecosystem
  - http://github.com/opencontainers/go-selinux

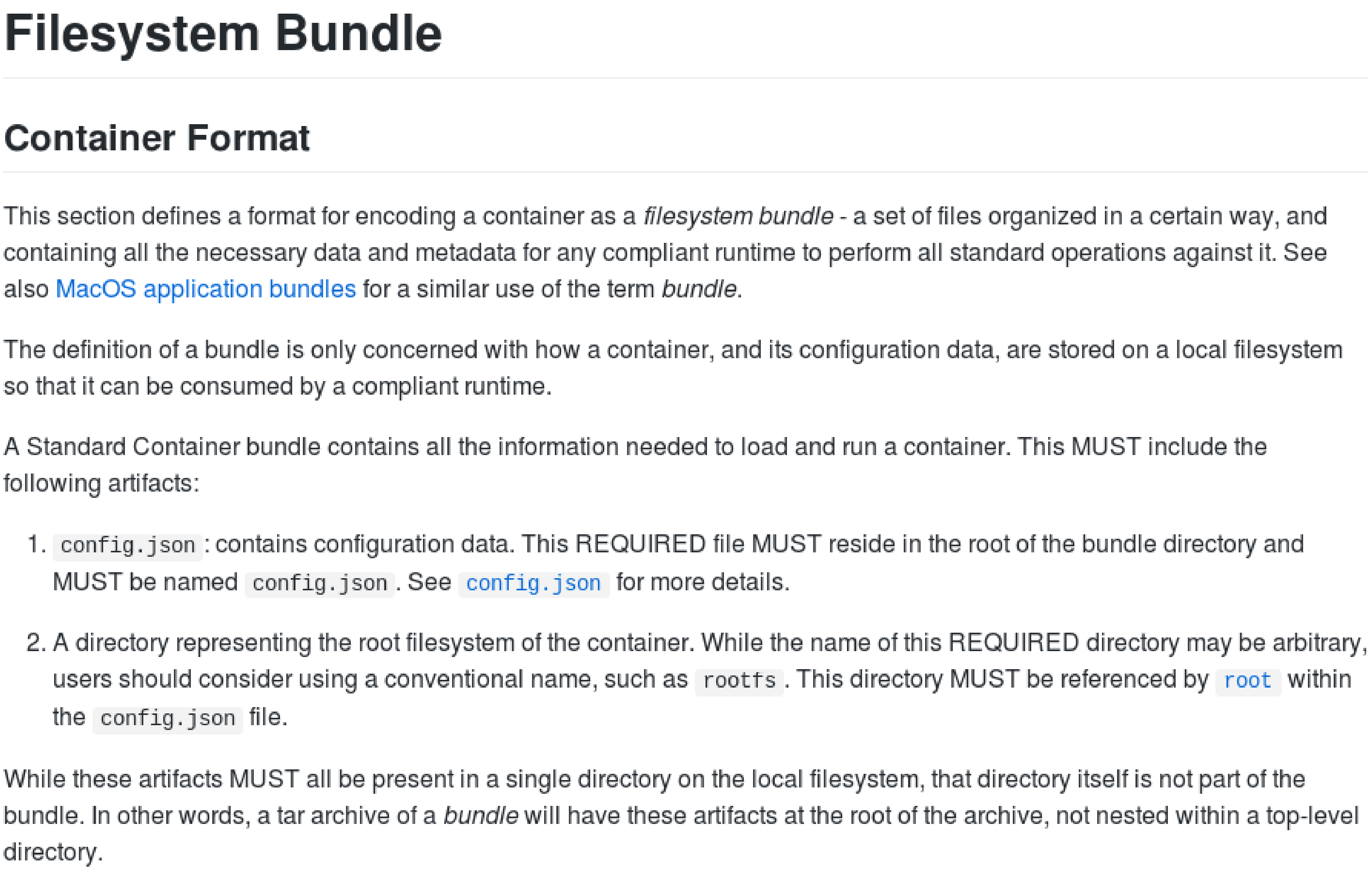
Runtime Spec

- Main Content



- Bundle Structure
  - A format for encoding a container
- Configuration
  - Including supported platforms and details the fields that enable the creation of a container
- Runtime & Lifecycle
  - Execution environment and actions of container lifecycle

Runtime Spec Screenshot



Filesystem Bundle

Container Format

This section defines a format for encoding a container as a *filesystem bundle* - a set of files organized in a certain way, and containing all the necessary data and metadata for any compliant runtime to perform all standard operations against it. See also MacOS application bundles for a similar use of the term *bundle*.

The definition of a bundle is only concerned with how a container, and its configuration data, are stored on a local filesystem so that it can be consumed by a compliant runtime.

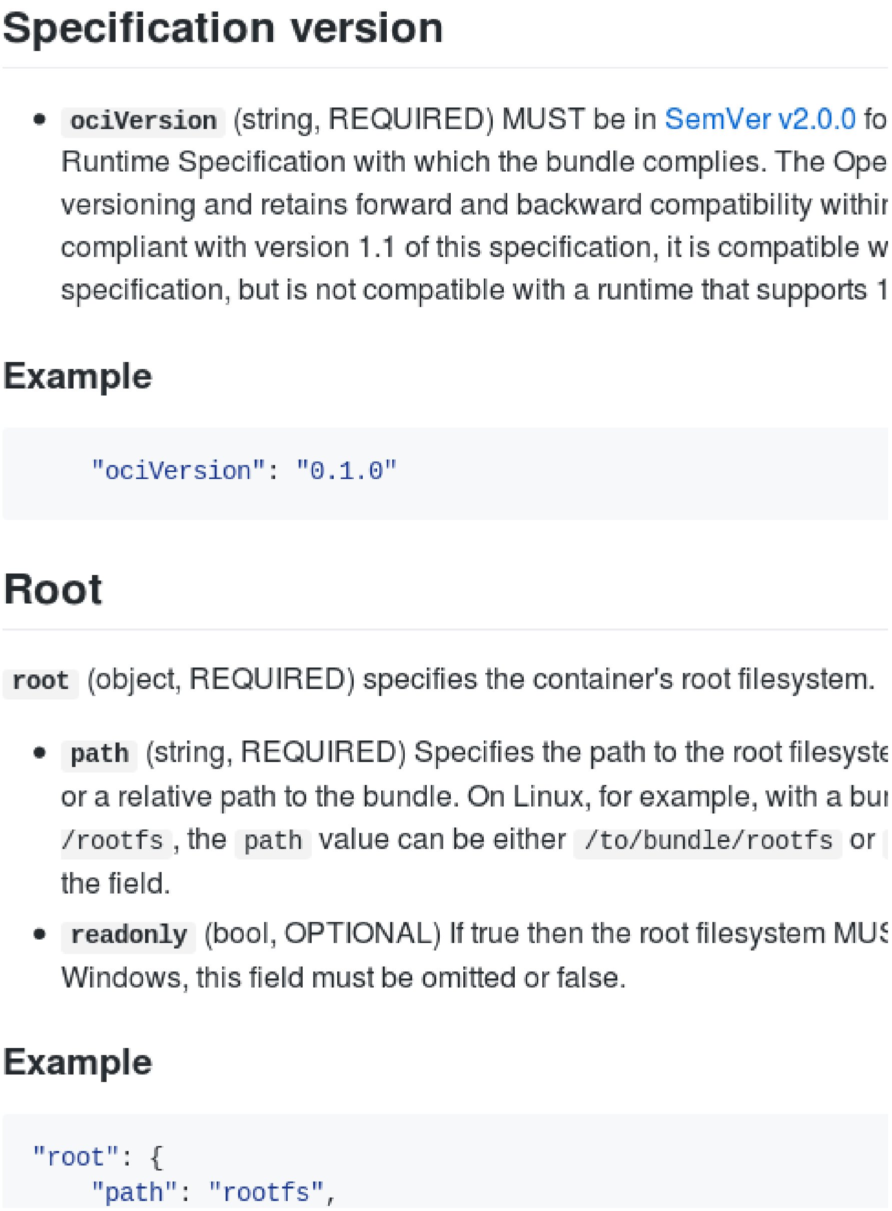
A Standard Container bundle contains all the information needed to load and run a container. This MUST include the following artifacts:

1. `config.json` : contains configuration data. This REQUIRED file MUST reside in the root of the bundle directory and MUST be named `config.json` . See `config.json` for more details.

2. A directory representing the root filesystem of the container. While the name of this REQUIRED directory may be arbitrary, users should consider using a conventional name, such as `rootfs` . This directory MUST be referenced by `root` within the `config.json` file.

While these artifacts MUST all be present in a single directory on the local filesystem, that directory itself is not part of the bundle. In other words, a tar archive of a *bundle* will have these artifacts at the root of the archive, not nested within a top-level directory.

Runtime Spec Screenshot



Specification version

- `ociVersion` (string, REQUIRED) MUST be in SemVer v2.0.0 fo Runtime Specification with which the bundle complies. The Ope versioning and retains forward and backward compatibility withir compliant with version 1.1 of this specification, it is compatible w specification, but is not compatible with a runtime that supports 1

Example

```
    "ociVersion": "0.1.0"
```

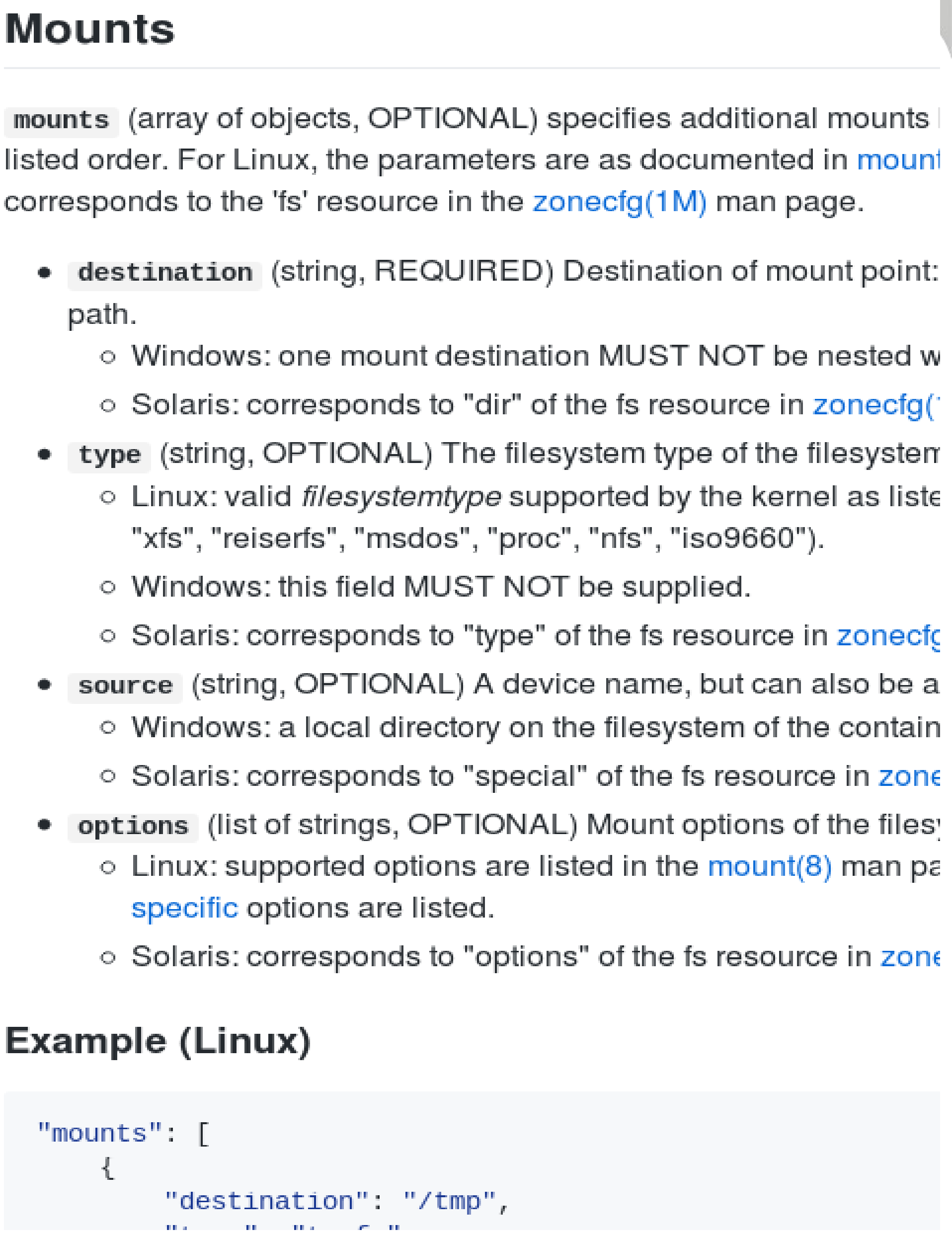
Root

`root` (object, REQUIRED) specifies the container's root filesystem.

- `path` (string, REQUIRED) Specifies the path to the root filesyste or a relative path to the bundle. On Linux, for example, with a bu `/rootfs`, the `path` value can be either `/to/bundle/rootfs` or the field.
- `readonly` (bool, OPTIONAL) If true then the root filesystem MUS Windows, this field must be omitted or false.

Example

```
"root": {
    "path": "rootfs",
```



Mounts

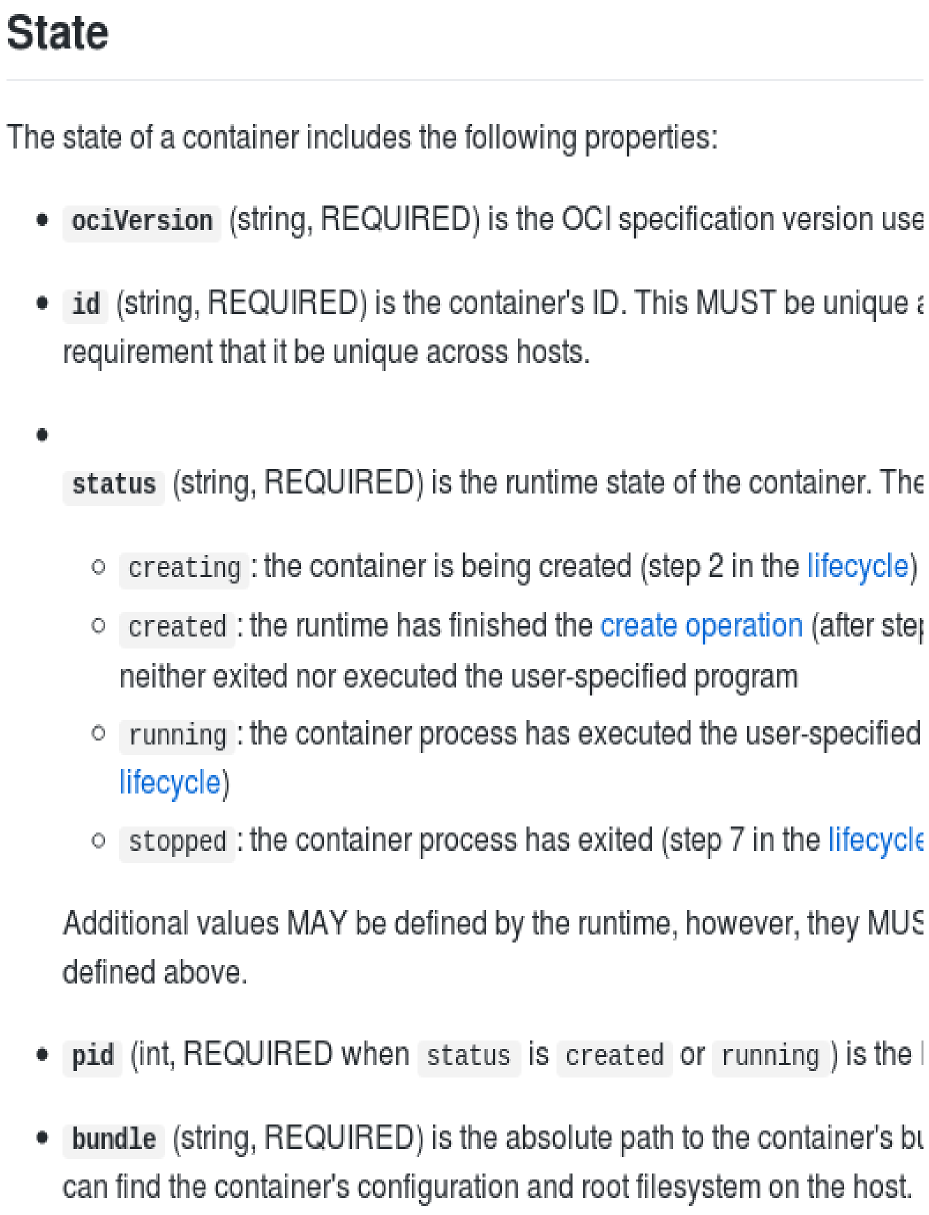
`mounts` (array of objects, OPTIONAL) specifies additional mounts listed order. For Linux, the parameters are as documented in mount corresponds to the 'fs' resource in the zonecfg(1M) man page.

- `destination` (string, REQUIRED) Destination of mount point: path.
  - Windows: one mount destination MUST NOT be nested w
  - Solaris: corresponds to "dir" of the fs resource in zonecfg(
- `type` (string, OPTIONAL) The filesystem type of the filesystem
  - Linux: valid *filesystemtype* supported by the kernel as liste "xfs", "reiserfs", "msdos", "proc", "nfs", "iso9660").
  - Windows: this field MUST NOT be supplied.
  - Solaris: corresponds to "type" of the fs resource in zonecf
- `source` (string, OPTIONAL) A device name, but can also be a
  - Windows: a local directory on the filesystem of the contain
  - Solaris: corresponds to "special" of the fs resource in zone
- `options` (list of strings, OPTIONAL) Mount options of the files
  - Linux: supported options are listed in the mount(8) man pa specific options are listed.
  - Solaris: corresponds to "options" of the fs resource in zone

Example (Linux)

```
"mounts": [
    {
        "destination": "/tmp",
```

Runtime Spec Screenshot

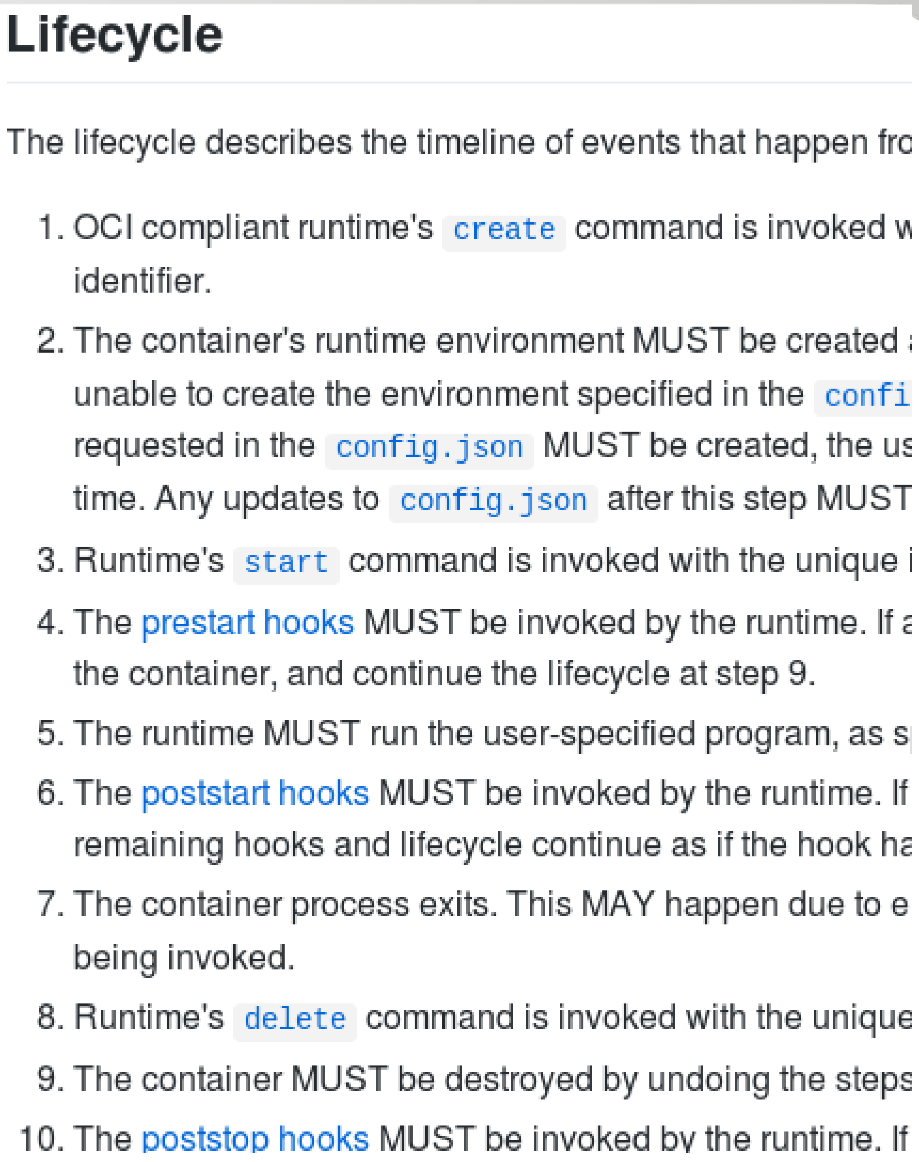


State

The state of a container includes the following properties:

- `ociVersion` (string, REQUIRED) is the OCI specification version use
- `id` (string, REQUIRED) is the container's ID. This MUST be unique a requirement that it be unique across hosts.
- `status` (string, REQUIRED) is the runtime state of the container. The
  - `creating` : the container is being created (step 2 in the lifecycle)
  - `created` : the runtime has finished the create operation (after ste neither exited nor executed the user-specified program
  - `running` : the container process has executed the user-specified lifecycle)
  - `stopped` : the container process has exited (step 7 in the lifecycle

  Additional values MAY be defined by the runtime, however, they MUS defined above.
- `pid` (int, REQUIRED when `status` is `created` or `running`) is the
- `bundle` (string, REQUIRED) is the absolute path to the container's bu can find the container's configuration and root filesystem on the host.

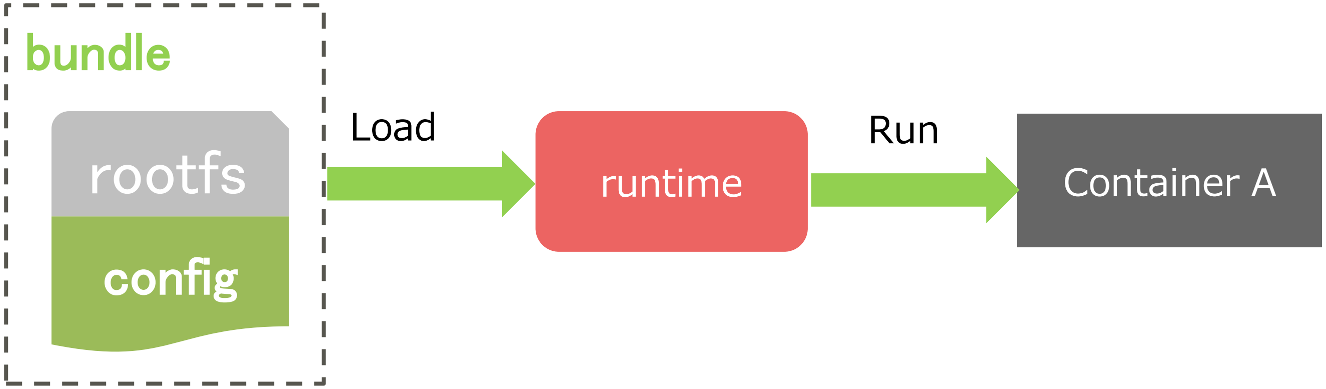


Lifecycle

The lifecycle describes the timeline of events that happen fro

1. OCI compliant runtime's `create` command is invoked w identifier.
2. The container's runtime environment MUST be created unable to create the environment specified in the `confi` requested in the `config.json` MUST be created, the us time. Any updates to `config.json` after this step MUST
3. Runtime's `start` command is invoked with the unique i
4. The prestart hooks MUST be invoked by the runtime. If a the container, and continue the lifecycle at step 9.
5. The runtime MUST run the user-specified program, as s
6. The poststart hooks MUST be invoked by the runtime. If remaining hooks and lifecycle continue as if the hook ha
7. The container process exits. This MAY happen due to e being invoked.
8. Runtime's `delete` command is invoked with the unique
9. The container MUST be destroyed by undoing the steps
10. The poststop hooks MUST be invoked by the runtime. If

Bundle & Container

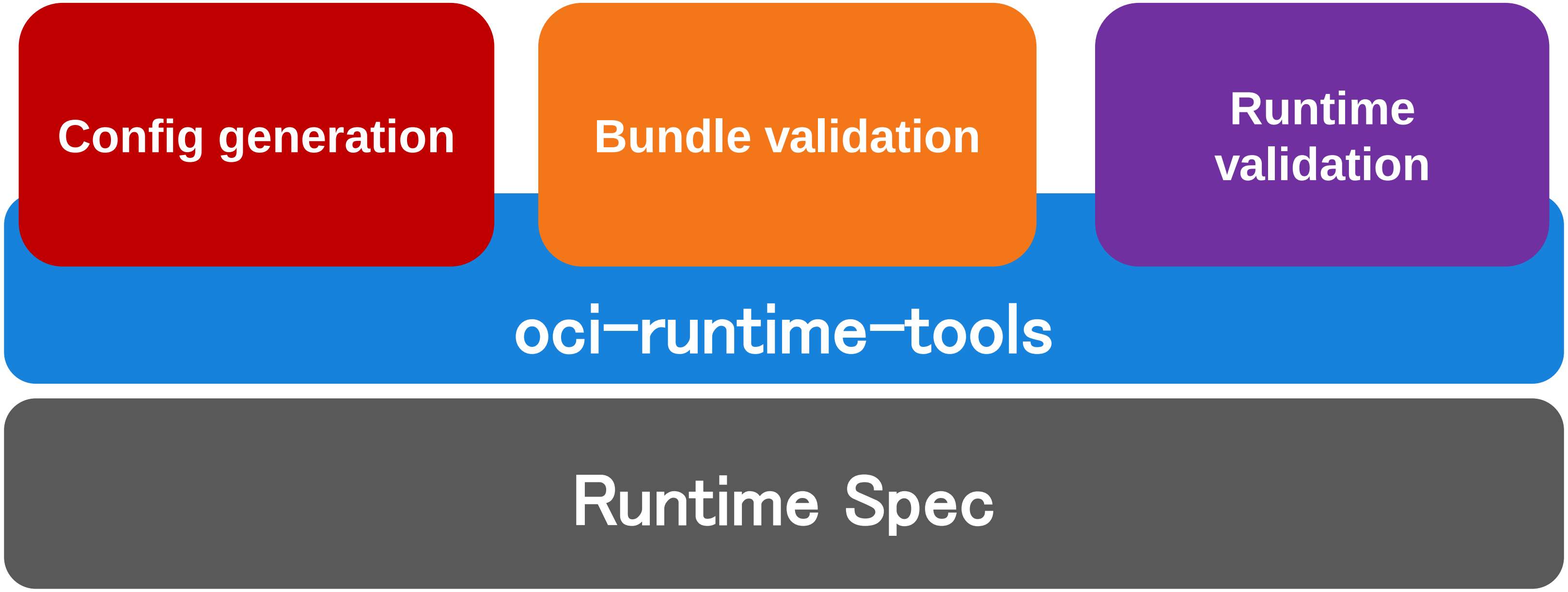


Why Runtime Tools

- How to judge if a bundle is usable and what’s the problem
- How to judge a container is portable
- How to judge if a runtime meets requirements of runtime spec

Runtime Tools

- Main Structure



Config generation

Bundle validation

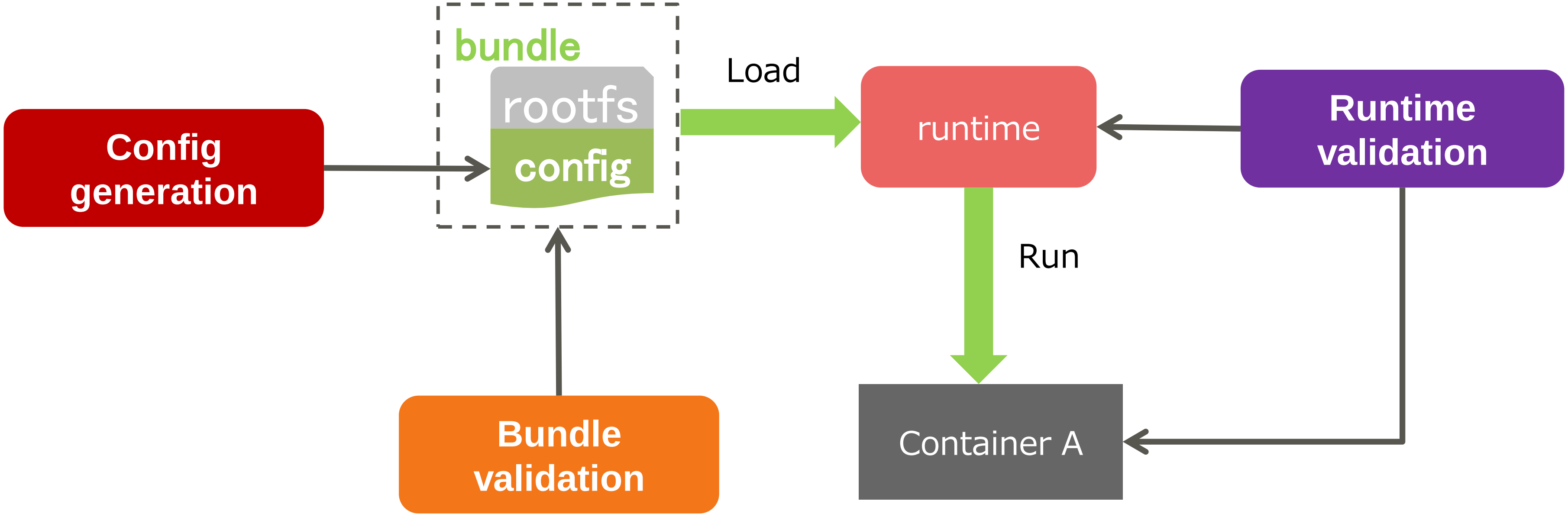
Runtime validation

oci-runtime-tools

Runtime Spec

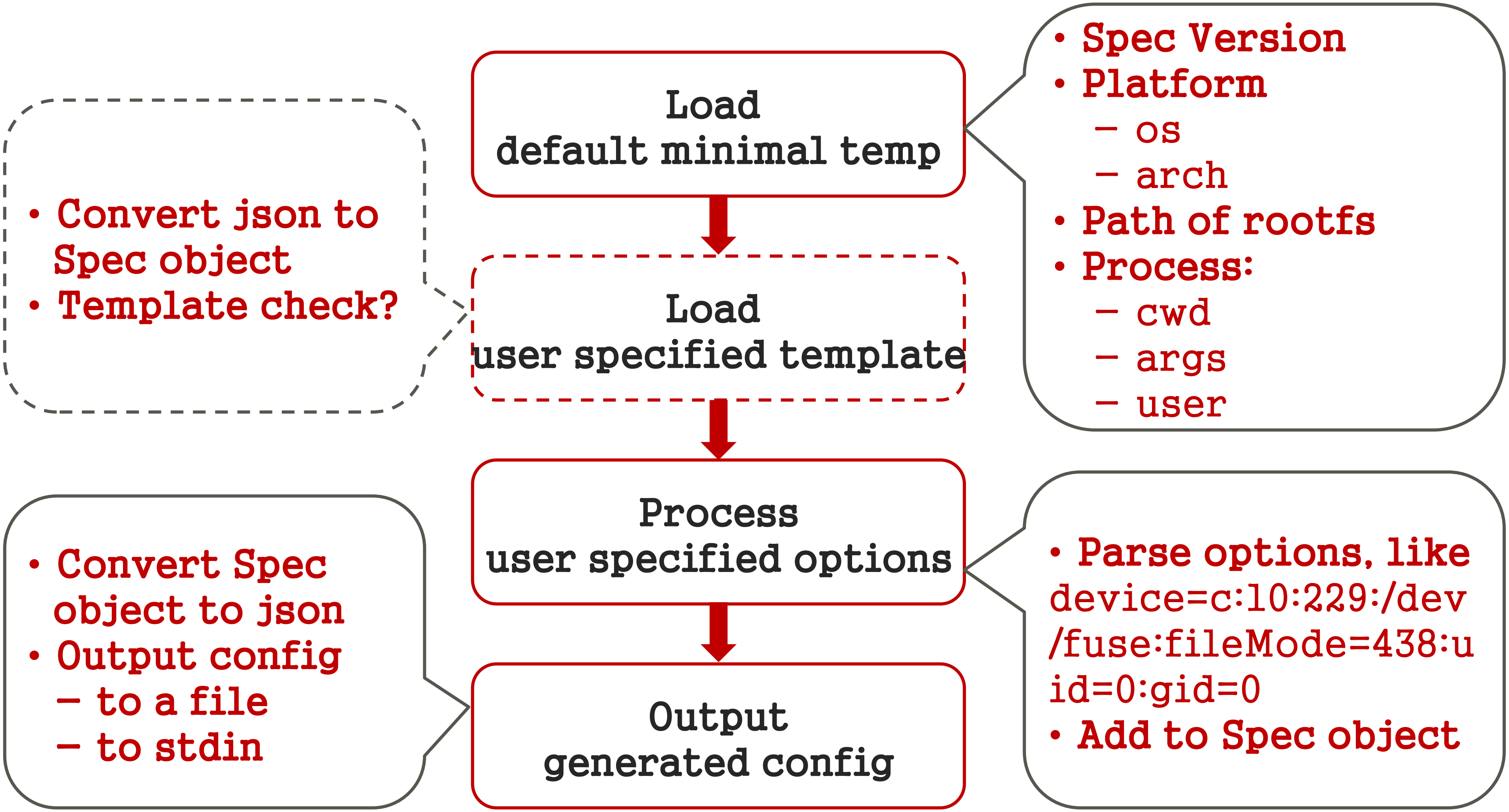
Runtime Tools

Usage & Relation



Configuration Generation

Work Flow



Load
default minimal temp

Load
user specified template

Process
user specified options

Output
generated config

- Spec Version
- Platform
  - os
  - arch
- Path of rootfs
- Process:
  - cwd
  - args
  - user

- Convert json to Spec object
- Template check?

- Parse options, like device=c:10:229:/dev/fuse:fileMode=438:uid=0:gid=0
- Add to Spec object

- Convert Spec object to json
- Output config
  - to a file
  - to stdin

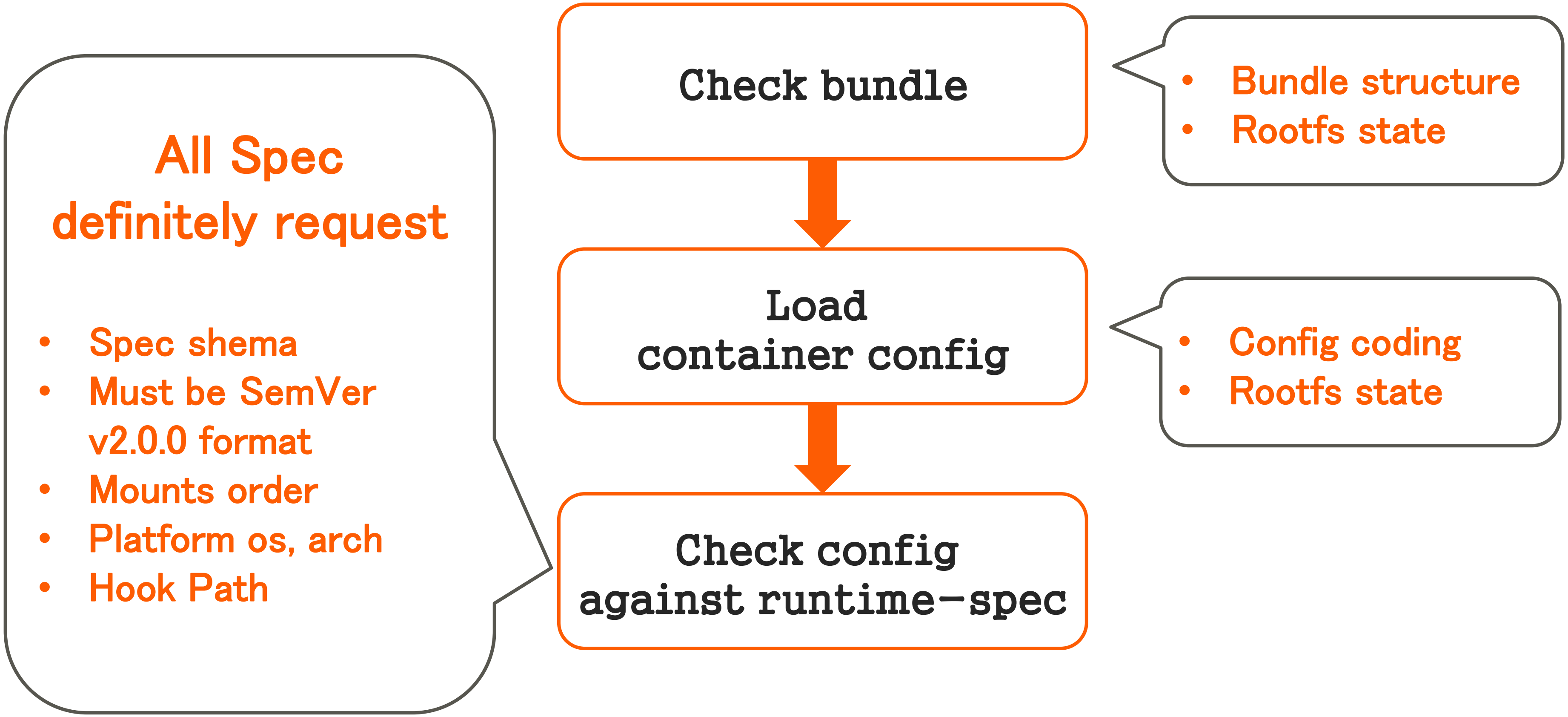
Configuration Generation

■ Example

```
$ oci-runtime-tool generate
{           ″ociVersion″: ″1.0.0-rc1-dev″,
            ″platform″: {
                        ″os″: ″linux″,
                        ″arch″: ″amd64″
            },
            ″process″: {
                        ″user″: {
                                    ″uid″: 0,
                                    ″gid″: 0
                        },
                        ″args″: [
                                    ″sh″
                        ],
                        ″env″: [
                                    ″PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin
```

Bundle Validation

Work Flow



Check bundle

- Bundle structure
- Rootfs state

Load container config

- Config coding
- Rootfs state

Check config against runtime-spec

All Spec definitely request

- Spec shema
- Must be SemVer v2.0.0 format
- Mounts order
- Platform os, arch
- Hook Path

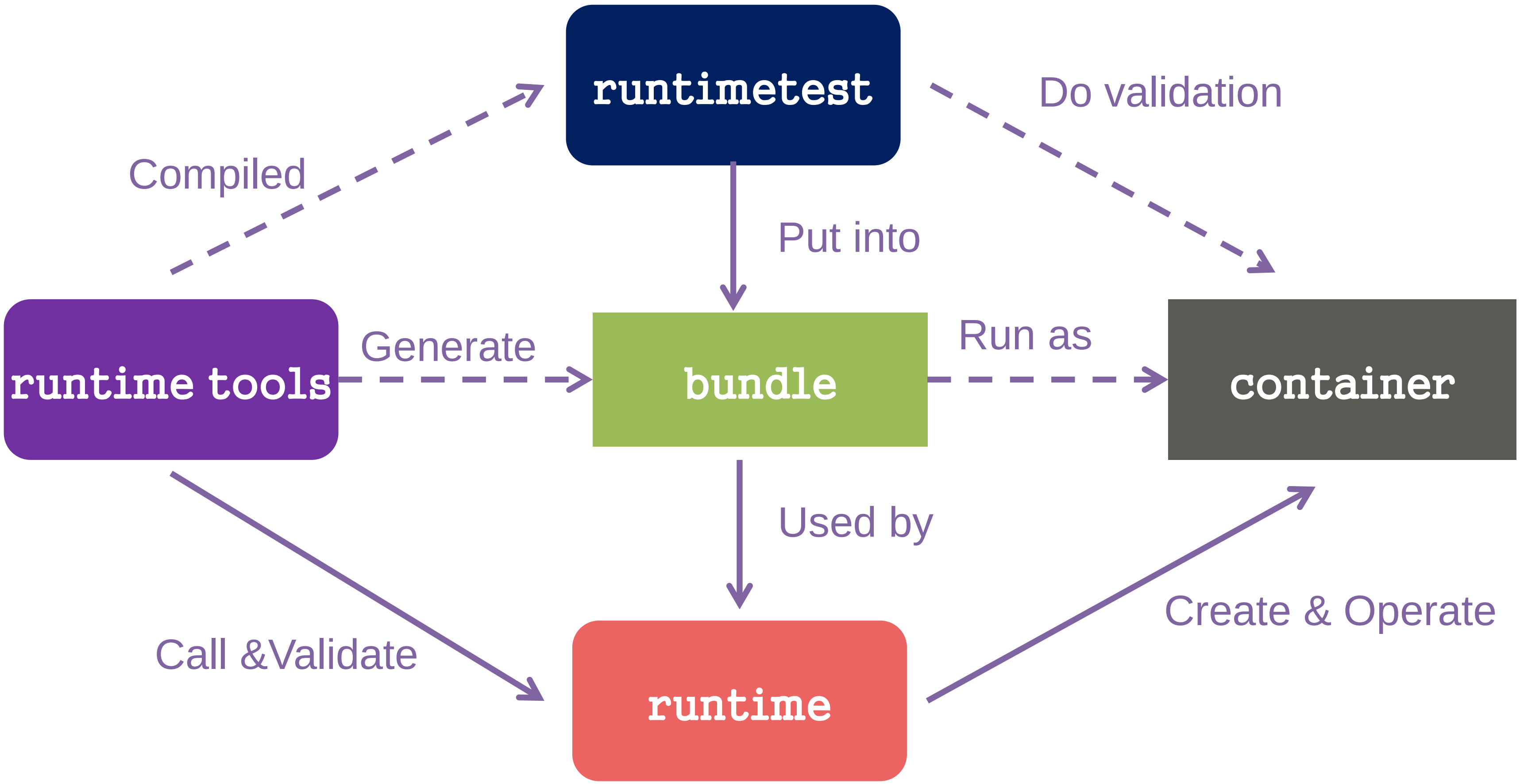
Bundle Validation

Example

```
$ oci-runtime-tool --host-specific --log-level=debug validate --path ~/testdir/
DEBU[0000] check rootfs path
DEBU[0000] check mandatory fields
DEBU[0000] check semver
DEBU[0000] check mounts
DEBU[0000] check platform
DEBU[0000] check process
DEBU[0000] check os
DEBU[0000] check linux
DEBU[0000] check linux resources
DEBU[0000] check linux seccomp
DEBU[0000] check hooks
Bundle validation succeeded.
```

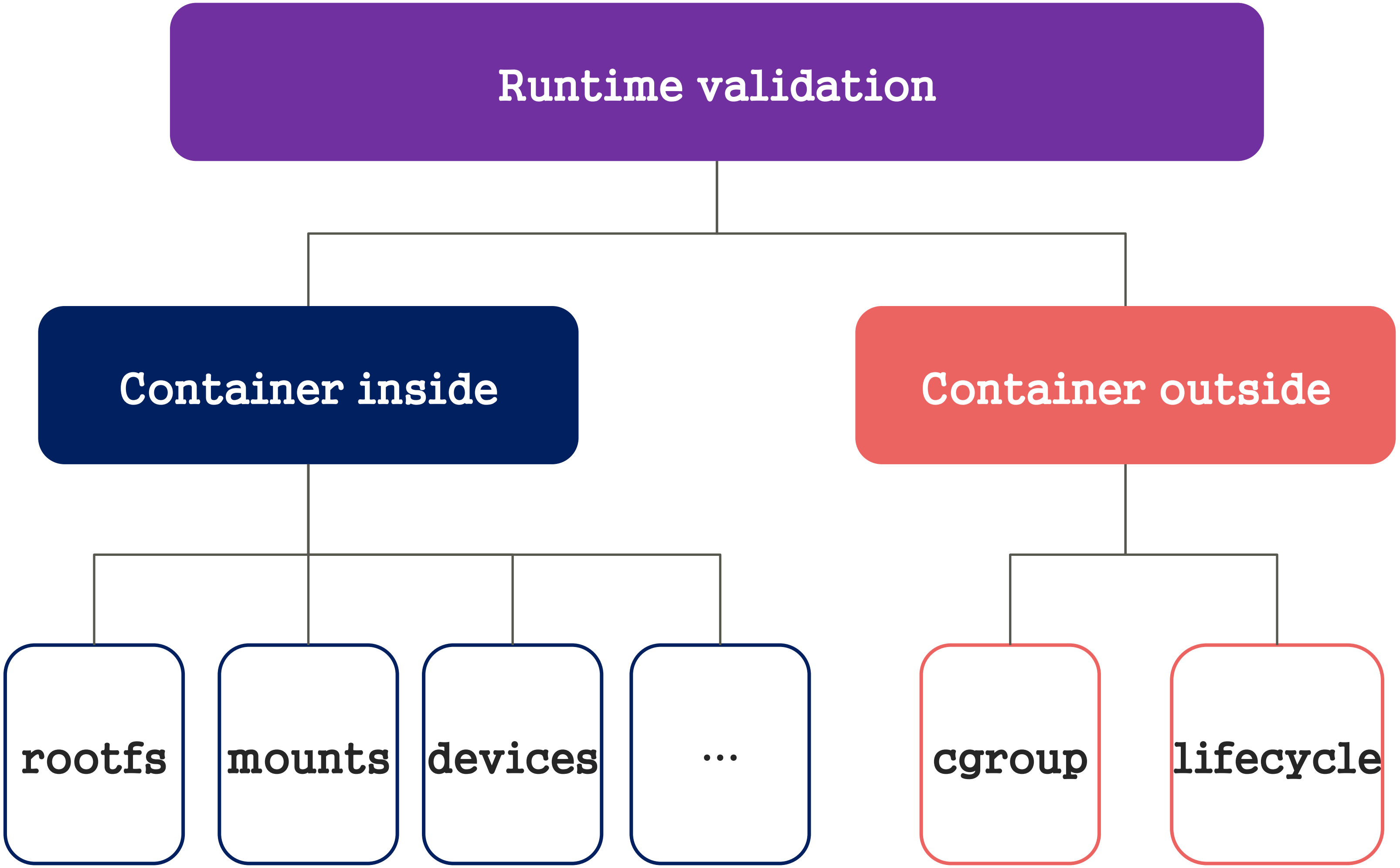
Runtime Validation

Work Flow



Runtime Validation

Detailed Validation



Runtime validation

Container inside

Container outside

rootfs

mounts

devices

…

cgroup

lifecycle

Runtime Validation

FUJITSU

Example

```
$ make localvalidation
RUNTIME=runc go test –tags ″″  –v github.com/opencontainers/runtime–tools/validation
=== RUN   TestValidateBasicTAP version 13
ok 1 – root filesystem
ok 2 – hostname
ok 3 – mounts
ok 4 – capabilities
ok 5 – default symlinks
ok 6 – default devices
ok 7 – linux devices
ok 8 – linux process
ok 9 – masked paths
ok 10 – oom score adj
ok 11 – read only paths
ok 12 – rlimits
ok 13 – sysctls
ok 14 – uid mappings
```

Current State of Runtime-tools

- Config Generation
  - already finished 80%
  - except blkio, hugepage, devices (but submitted patch)
- Bundle Validation
  - already finished 70%
  - except groups related, schema
- Runtime Validation
  - reforming (container inside almost finished)
  - cgroups, lifecycle

Our Contribution

- Runtime-tools: 117 commits
  - Improve coverage of bundle & runtime tests
  - Enhance functionality of runtime-spec generation
  - Bug fix for code and document
- Runtime-spec: 54 commits
  - Help specify spec entries
- Image-tools: 27 commits
  - Enhance functionality
  - Bug fix
- Image-spec: 24 commits
  - Help specify spec entries

Future Plans

- Order of options problem
  - oci-runtime-tools generate --rlimits-remove-all --rlimits-add RLIMIT_NOFILE:10:10
- Runtime validation improvement
  - cgroup related validation
  - container lifecycle validation
- Platform portability
  - currently can only work on Linux
  - cross validation, windows bundle on Linux?
- More tests and trials by runtime creators

Q&A

Thank you!
Q&A

FUJITSU

shaping tomorrow with you