Secrets Management in Mesos

Vinod Kone (*vinodkone@apache.org*)

MesosCon EU 2017

About me

- Apache Mesos PMC and Committer
- Engineering Manager for Mesos team @ Mesosphere
- Previously Tech Lead for Mesos team @ Twitter
- PhD in Computer Science @ University of California Santa Barbara

What is a secret?

- Any sensitive information
  - Passwords
  - SSH Keys
  - Certificates
  - API Keys

- Secrets should only be visible to authorized users
  - Typically only to the owner of the secret

How should we handle secrets?

- Time in transit should be minimized
- Avoid persisting to disk if possible
- Limit possibility of interception

Use case #1: Image pull secrets

- How to download images from a private Docker registry?
  - Needs credentials to authenticate

Existing Solutions	**Limitations**
Docker Containerizer • Registry 1.0: Add .dockercfg as a TaskInfo URI. $HOME is set to $MESOS_SANDBOX • Registry 2.0: Add docker.tar.gz as a TaskInfo URI. Archive should contain .docker/config.json	• URIs accessible to all tasks / users • Credentials are downloaded to sandbox => visible on host fs even after container terminates

Use case #1: Image pull secrets

- How to download images from a private Docker registry?
  - Needs credentials to authenticate

Existing Solutions	Limitations
Docker Containerizer ● Registry 1.0: Add .dockercfg as a TaskInfo URI. $HOME is set to $MESOS_SANDBOX ● Registry 2.0: Add docker.tar.gz as a TaskInfo URI. Archive should contain .docker/config.json	● URIs accessible to all tasks / users ● Credentials are downloaded to sandbox => visible on host fs
Mesos Containerizer ● Add docker credentials to each agent via --docker_config flag	● Credentials need to be configured by operators and not application developers ● Per task credentials are not supported

Use case #2: Application secrets

- An application (Mesos task) needs access to credentials to talk to other services

Existing Solutions	**Limitations**
Pass secrets via \`data\` or \`labels\` in TaskInfo	● Labels exposed in API endpoints ● TaskInfo is visible on network without SSL

Use case #2: Application secrets

- An application (Mesos task) needs access to credentials to talk to other services

Existing Solutions	**Limitations**
Pass secrets via \`data\` or \`labels\` in TaskInfo	● Labels exposed in API endpoints ● TaskInfo is visible on network without SSL
Fetch secrets from URIs	● No support for authenticated URIs ● Downloaded to sandbox => visible on host fs even after container termination

Use case #2: Application secrets

- An application (Mesos task) needs access to credentials to talk to other services

Existing Solutions	Limitations
Pass secrets via \`data\` or \`labels\` in TaskInfo	● Labels exposed in API endpoints ● TaskInfo is visible on network without SSL
Fetch secrets from URIs	● No support for authenticated URIs ● Downloaded to sandbox => visible on host fs
Out of band mechanisms (hooks, isolator modules)	● Complicated ● Not reusable

Use case #3: Executor authentication

- Executors need to authenticate with agents with unique credentials
    - Credentials need to be securely passed to the executor

Use case #3: Executor authentication

- Executors need to authenticate with agents with unique credentials
  - Credentials need to be securely passed to the executor

- There is historically no native support for executor authentication
  - Neither in v0 or v1 APIs
  - Tasks can spoof as executors!

Goals

- Add first class support for Secrets in Mesos
- Integrate with 3rd party secret stores (e.g., HashiCorp Vault)
- Support environment based and file based secrets

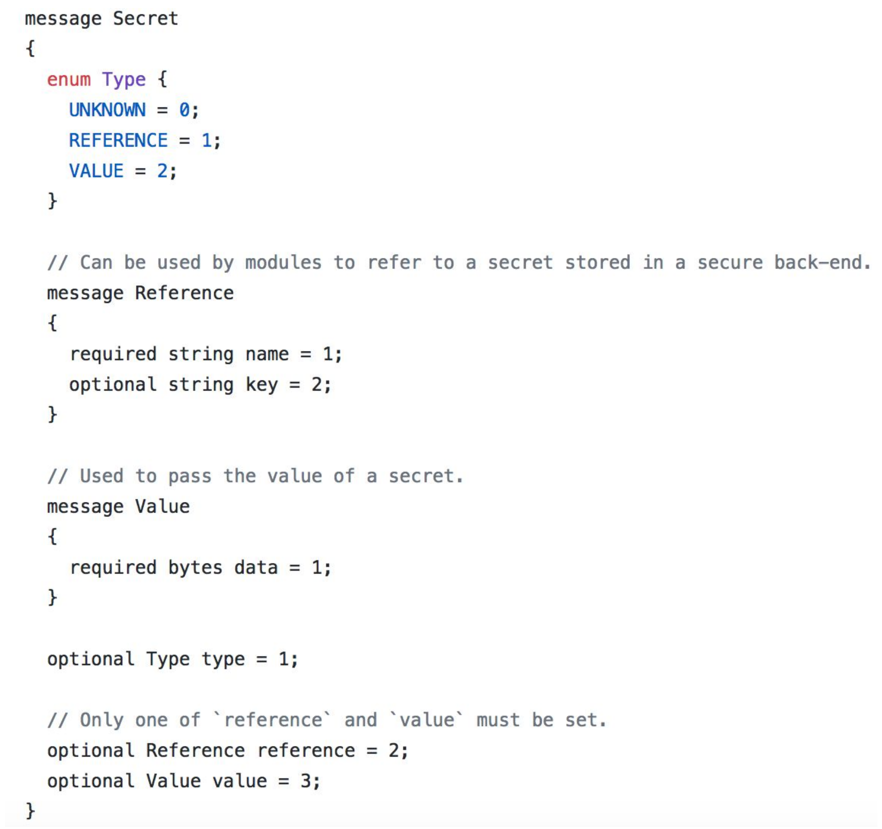
Solution overview

- Secret

- Secret Resolver

- Secret Isolators
  - `environment_secret`
  - `volume/secret`

Secret Protobuf



```
message Secret
{
  enum Type {
    UNKNOWN = 0;
    REFERENCE = 1;
    VALUE = 2;
  }

  // Can be used by modules to refer to a secret stored in a secure back-end.
  message Reference
  {
    required string name = 1;
    optional string key = 2;
  }

  // Used to pass the value of a secret.
  message Value
  {
    required bytes data = 1;
  }

  optional Type type = 1;

  // Only one of `reference` and `value` must be set.
  optional Reference reference = 2;
  optional Value value = 3;
}
```

Secret Resolver Interface

```
class SecretResolver
{
public:
  // Factory method used to create a SecretResolver instance. If the
  // `name` parameter is provided, the module is instantiated
  // using the `ModuleManager`. Otherwise, a "default" secret resolver
  // instance (defined in `src/secret/resolver.hpp`) is returned.
  static Try<SecretResolver*> create(const Option<std::string>& name = None());

  virtual ~SecretResolver() {}

  // Validates the given secret, resolves the secret reference (by potentially
  // querying a secret backend store), and returns the data associated with
  // the secret.
  virtual process::Future<Secret::Value> resolve(
      const Secret& secret) const = 0;

protected:
  SecretResolver() {}
};
```

Architecture

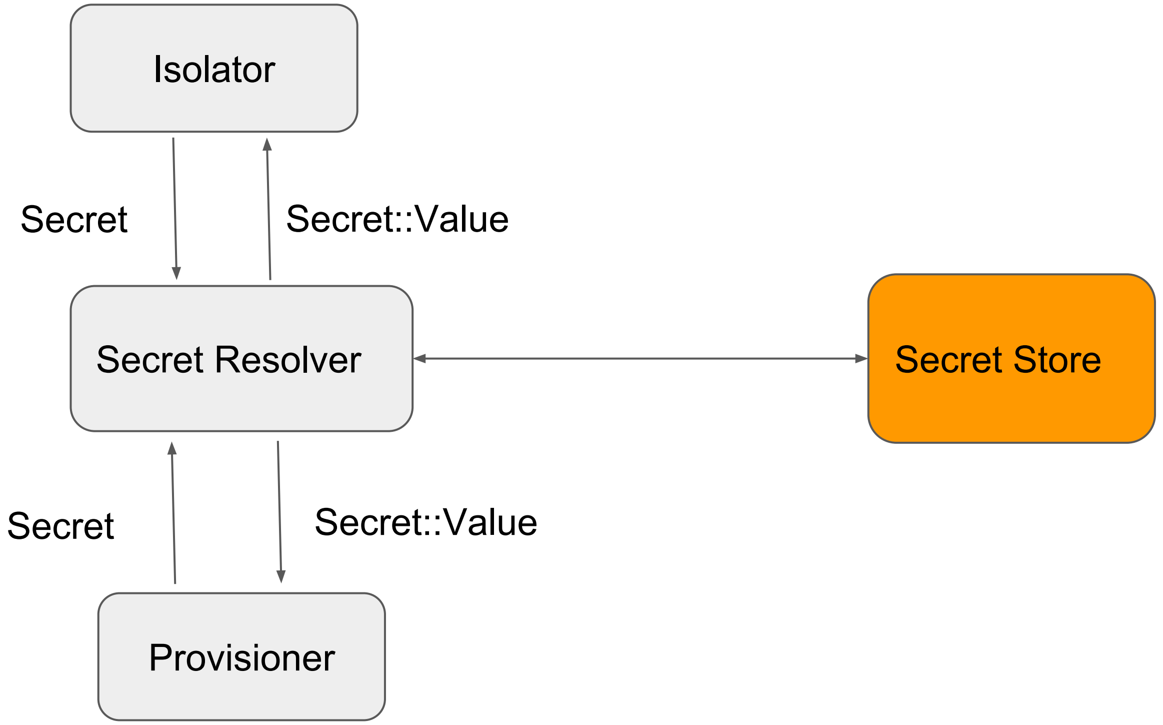
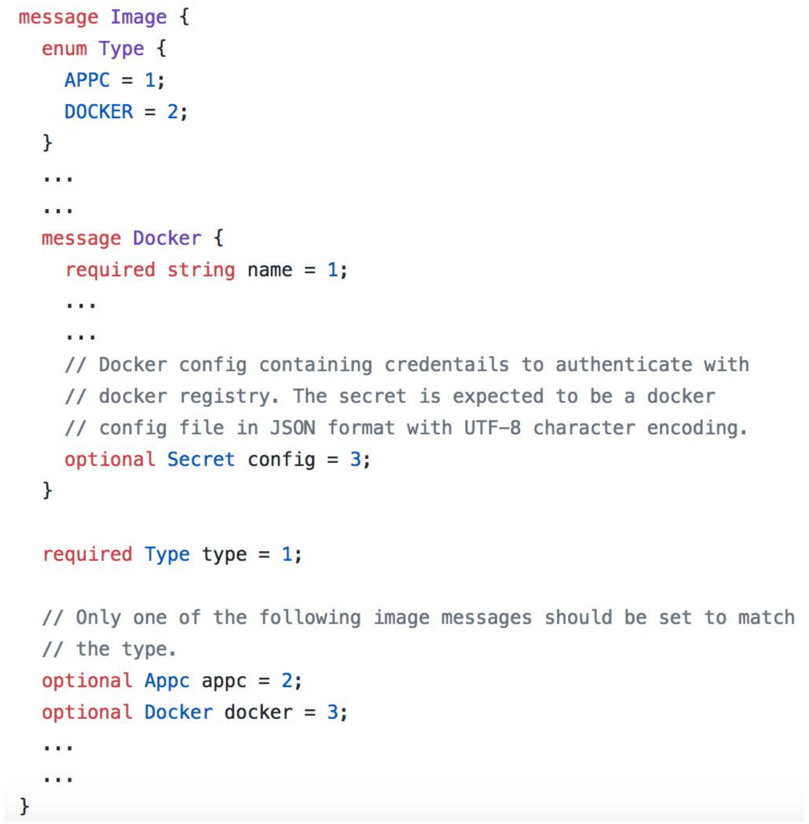
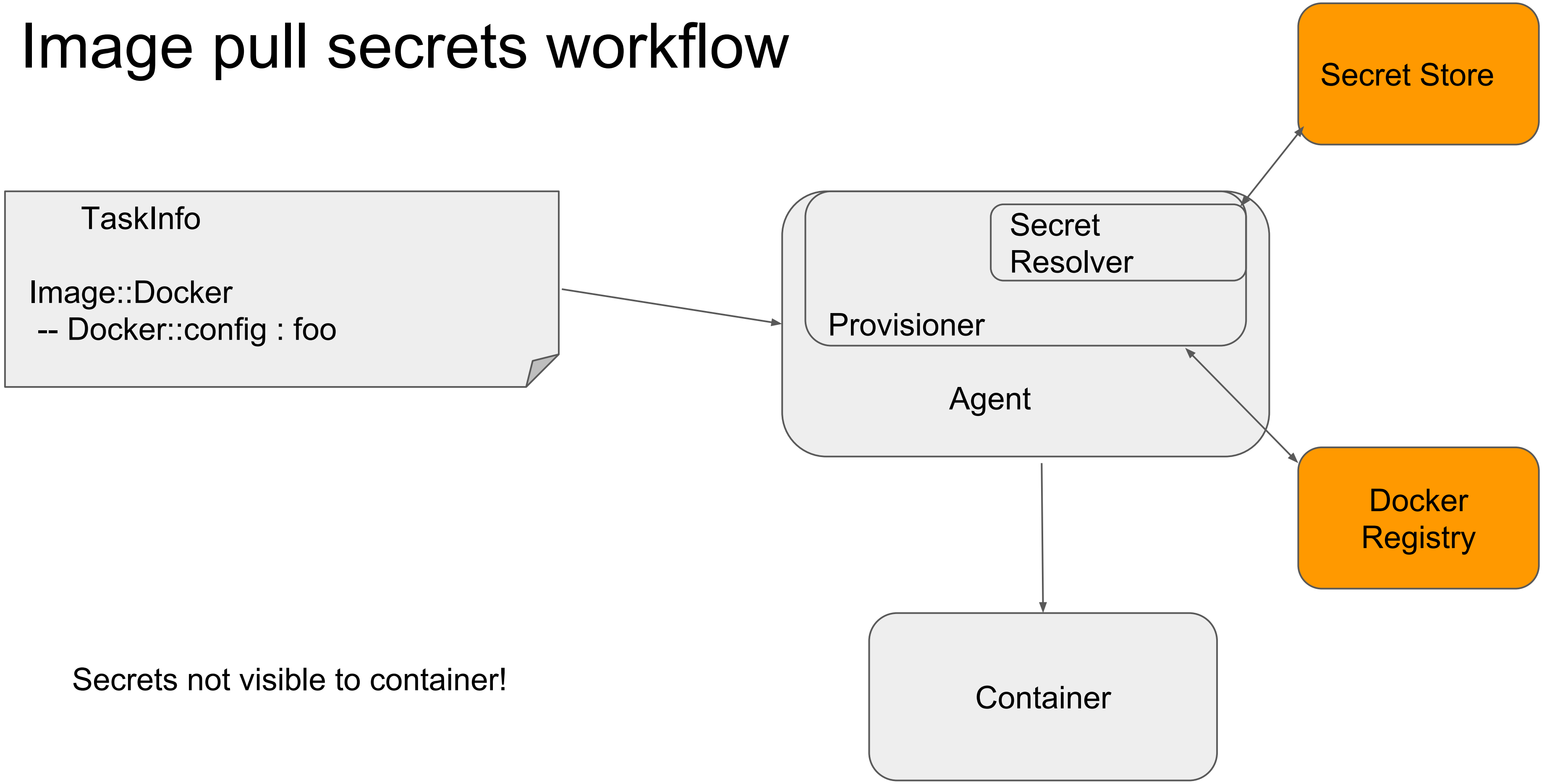


Image pull secrets



```
message Image {
  enum Type {
    APPC = 1;
    DOCKER = 2;
  }
  ...
  ...
  message Docker {
    required string name = 1;
    ...
    ...
    // Docker config containing credentails to authenticate with
    // docker registry. The secret is expected to be a docker
    // config file in JSON format with UTF-8 character encoding.
    optional Secret config = 3;
  }

  required Type type = 1;

  // Only one of the following image messages should be set to match
  // the type.
  optional Appc appc = 2;
  optional Docker docker = 3;
  ...
  ...
}
```

Image pull secrets workflow

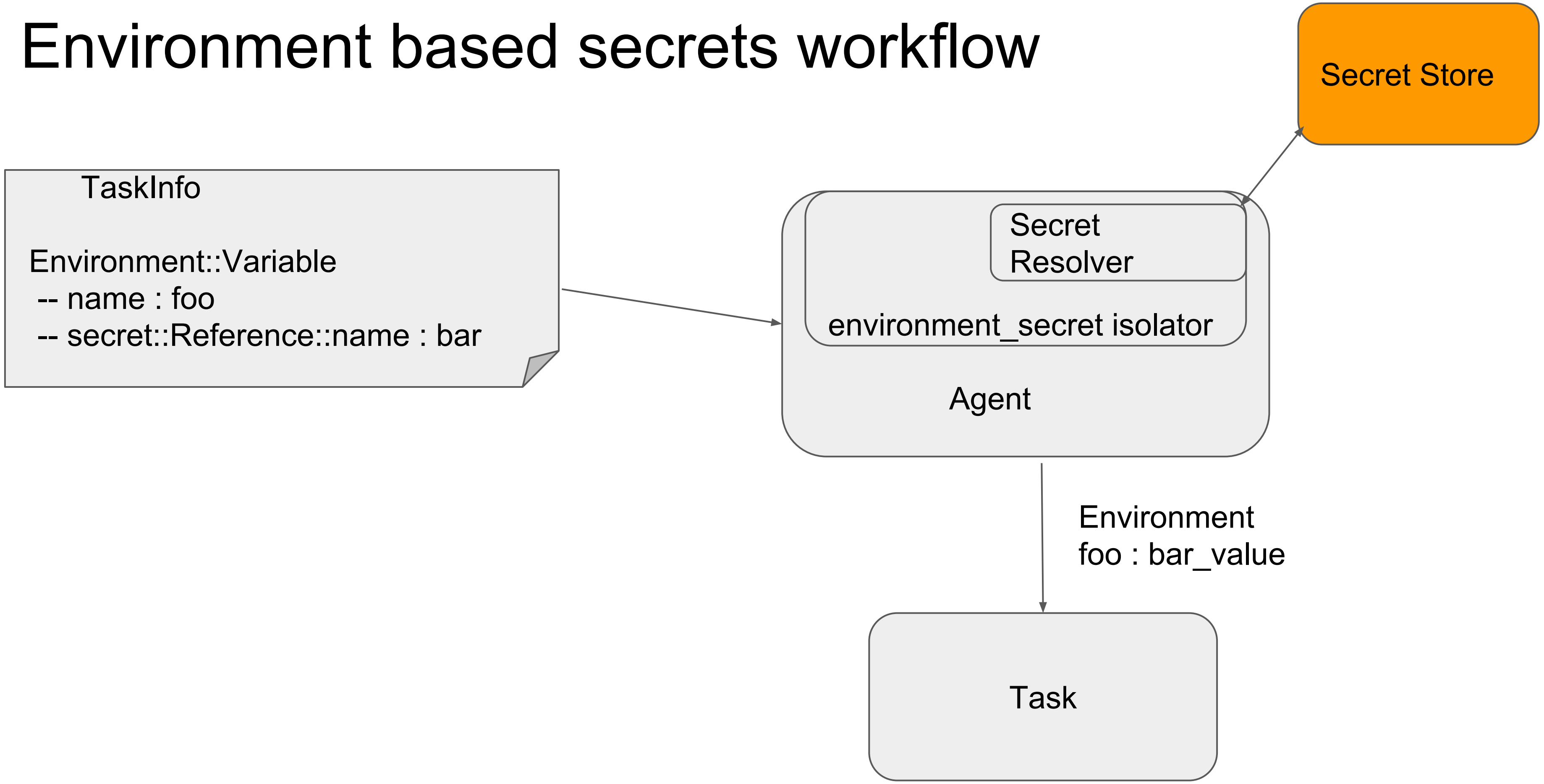


Secrets not visible to container!

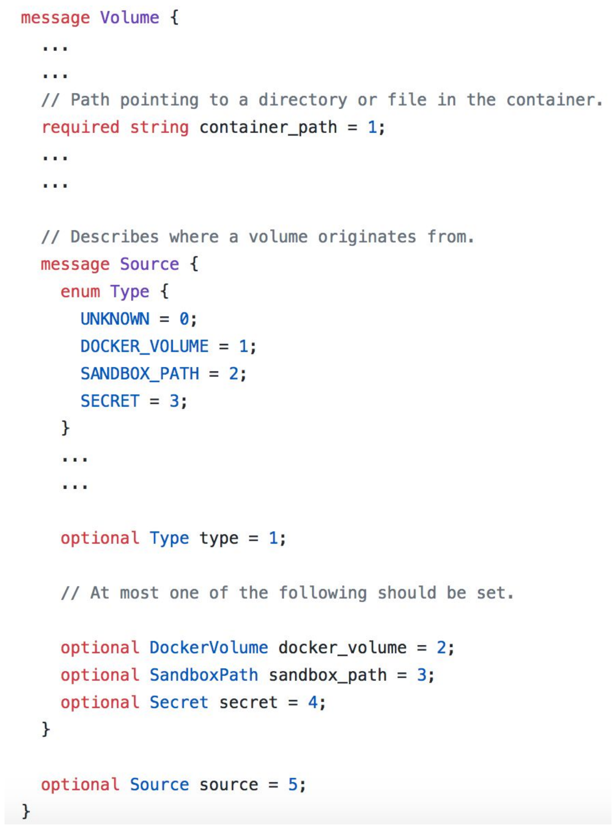
Environment based secrets

```
message Environment {
  message Variable {
    required string name = 1;

    enum Type {
      UNKNOWN = 0;
      VALUE = 1;
      SECRET = 2;
    }

    optional Type type = 3 [default = VALUE];

    // Only one of `value` and `secret` must be set.
    optional string value = 2;
    optional Secret secret = 4;
  }

  repeated Variable variables = 1;
}
```

Environment based secrets workflow

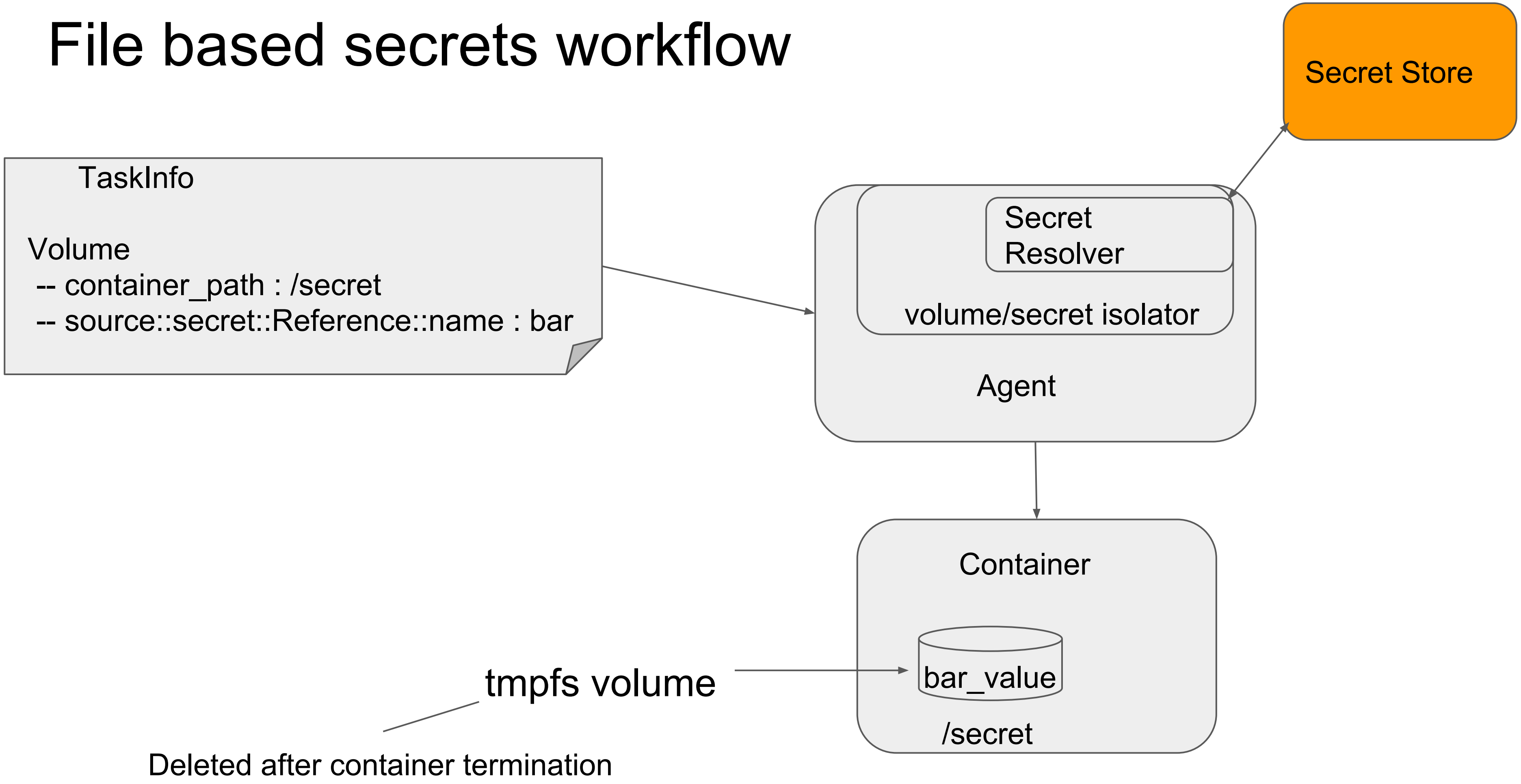


File based secrets



```
message Volume {
  ...
  ...
  // Path pointing to a directory or file in the container.
  required string container_path = 1;
  ...
  ...

  // Describes where a volume originates from.
  message Source {
    enum Type {
      UNKNOWN = 0;
      DOCKER_VOLUME = 1;
      SANDBOX_PATH = 2;
      SECRET = 3;
    }
    ...
    ...

    optional Type type = 1;

    // At most one of the following should be set.

    optional DockerVolume docker_volume = 2;
    optional SandboxPath sandbox_path = 3;
    optional Secret secret = 4;
  }

  optional Source source = 5;
}
```

File based secrets workflow



Feature Status

- Secrets support included in Mesos 1.3.0
  - Mesos Containerizer support for Image pull secrets
  - Environment based secrets
  - File based secrets

- Secret Resolver
  - Interface is modularized
  - `Value` based resolver included in Mesos repo
  - `Reference` based resolver can be implemented as a module

Demo

Future Work

- Image pull secrets
    - Support for Docker Containerizer
    - AppC / OCI support for Mesos Containerizer

- URI fetching
    - Use secrets to fetch URIs that require authentication
    - Fetch https URIs with TLS/SSL certificates

Acknowledgements

- Gilbert Song
- Kapil Arya
- Jie Yu
- Chun-Hung Hsiao

Thanks

Design docs: Image pull secrets, File based secrets, Executor authentication