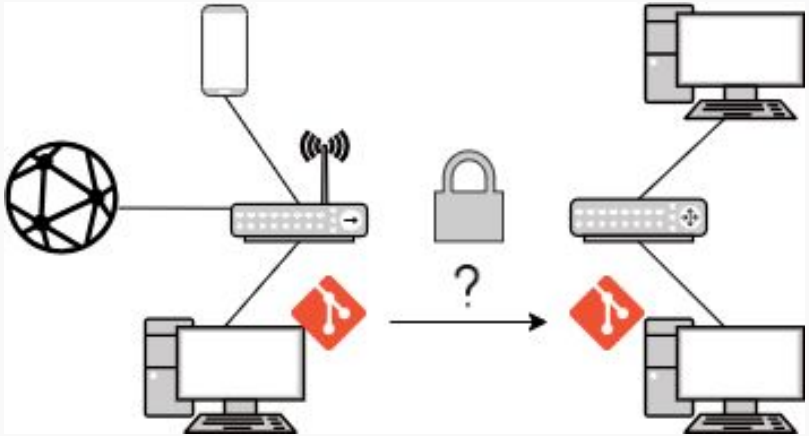
Gitoci

Git remote helper for storing repositories in OCI

By: Nathan Joslin
Advisor: Dr. Kyle Tarplee

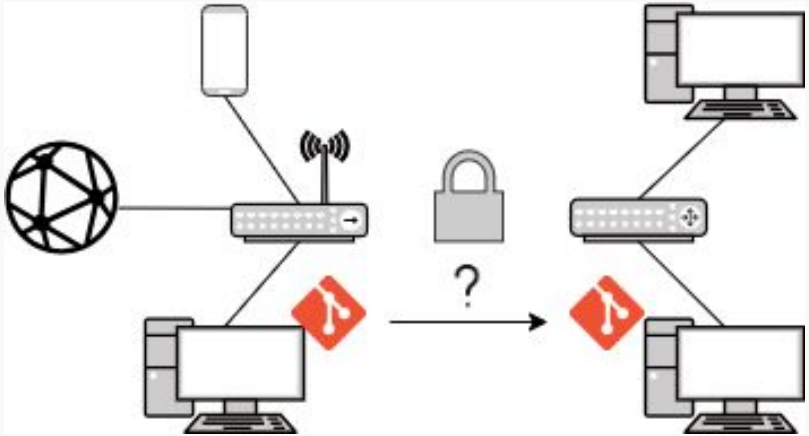
Problem Description

- Need for moving Git repos across air-gapped environments
- Core problems:
  - Efficient data transfer
  - Efficient data storage
  - Minimal maintenance
- Existing tools do not solve all three well
- How can we achieve the main goal by using their strengths?



Problem Description

- Git bundles
  - Good for offline transfers
- ORAS
  - Efficient transfers of OCI artifacts
- How can we transfer Git repositories to/from OCI registries?



Existing Solutions

Zarf

- CD for air gapped systems
- Large tool and “ecosystem”
- Lacks incremental updates
- Lacks git-lfs support

ORAS

- Efficient transfers of OCI artifacts
- Go package
- Command line interface
- CNCF Sandbox project

Git Bundles

- Native to git
- Intended for offline transfers
- “Thin” packs
- Difficult to manage at scale, large and/or many repositories
- Lacks git-lfs support

Proof-of-Concept

ASCE Data Tool:

- Prototype OCI data model using Git bundles
- Existing user base

```
$ export IMAGE_NAME=torvalids-linux-$(uuidgen)

$ ace-dt git to-oci git@github.com:torvalds/linux.git ttl.sh/${IMAGE_NAME}:1h

$ ace-dt git from-oci ttl.sh${IMAGE_NAME} git@github.com:user-example.git
```

Pros:

- Supports incremental updates
- Supports updating subsets of all repository references
- All work is done client side
- Supports git-lfs

Cons:

- Room for optimizations
- Subcommand of a larger tool
- Does not support native Git usage

Proposed Solution

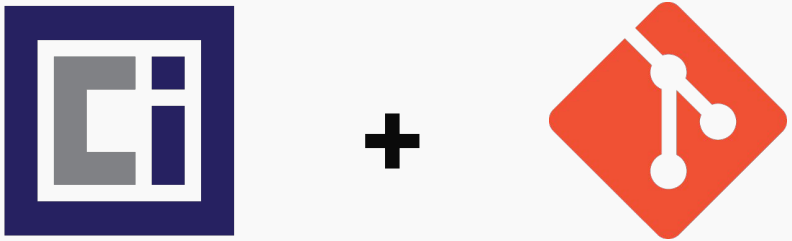
OCI Spec for Git Repositories + Git Remote Helper

Why OCI?

- An industry standard for image storage and transfer
- Increasing usage for alternative artifacts
- Take advantage of existing transfer tools

Why Git Remote Helper?

- Allow Git to interface with alternative repo formats
- Use git natively, or with existing tools (e.g. VSCode)
- Well-defined requirements



Proposed Solution

- OCI data model using Git packfiles
- Git remote helper to facilitate pushing and fetching to/from OCI
- All work done client side

```
# Clone directly from OCI registry
$ git clone oci://example.registry.com/repository/name:tag

# Add an OCI registry as a remote
$ git remote add <remote-name> oci://example.registry.com/repository/name:tag
```

```
# Fetch from a remote OCI registry
$ git fetch [<repository>] [<refspec>...]

# Push to a remote OCI registry
$ git push [<repository>] [<refspec>...]
```

Goals & Implementation

Goals:

- OCI Spec for Git Repos using packfiles
- Git Remote Helper
- Support for git-lfs
- Minimize overhead compared to native Git
- 50% test coverage

Implementation Details:

- Written in Go
- Key External Dependencies:
  - ORAS
  - Sourcegraph/conc
  - Go-git

Remote Helper Capabilities

Pushing

Capability	Support
connect	Yes
stateless-connect	No
push	Maybe
export	Maybe
no-private-update	No

Fetching

Capability	Support
connect	Yes
stateless-connect	No
fetch	Maybe
import	Maybe
check-connectivity	Maybe
get	Maybe

Misc.

Capability	Support
option	Maybe
refspec	Maybe
bidi-import	Maybe
export-marks	Maybe
import-marks	Maybe
signed-tags	Maybe
object-format	Maybe

Capabilities: Pushing

- Connect (Yes)
  - Utilizes Git's packfile protocol.
  - Bundles are an extension of packfiles, used by proof-of-concept.
  - Necessary for efficient pushes, and OCI storage.
- Stateless-Connect (No)
  - Described as experimental and intended for internal use.
- Push (Maybe)
  - Used to update remote history and refs with local objects.
  - Packfile protocol supported by *connect* is better suited for goals.
  - *Connect* is preferred, unless *push* or *export* is requested by remote helper.
  - May improve efficiency in certain situations, experimentation needed.

Capabilities: Pushing

- Export (Maybe)
  - Used to push objects over a fast-import stream.
  - Git prefers *push* over *export*.
  - More research needed.
- No-private-update (No)
  - Adds support for disabling private namespace updates.
  - Not within scope of project.
- See Capabilities for Pushing

Capabilities: Fetching

- Connect (Yes)
  - Utilizes Git's packfile protocol.
  - Bundles are an extension of packfiles, used by proof-of-concept.
  - Necessary for efficient fetches, and OCI storage.
- Stateless-Connect (No)
  - Described as experimental and intended for internal use.
- Fetch (Maybe)
  - Used to update local history and refs with remote objects.
  - Packfile protocol supported by *connect* is better suited for goals.
  - May improve efficiency in certain situations, experimentation needed.

Capabilities: Fetching

- Import (Maybe)
  - Used to fetch git objects over fast-import steam from a remote.
  - Maybe be useful to receive packfiles from OCI, and send objects to git.
  - More research necessary.
- Check-connectivity (Maybe)
  - Used to validate that a packfile, from a clone, i self contained.
  - Not necessary, but potentially useful.
- Get (Maybe)
  - Used to fetch files from a URI.
  - Not necessary for MVP, but potentially useful.
- See Fetching Capabilities

Capabilities: Miscellaneous

- Option (Maybe)
  - Potentially helpful for controlling verbosity and history depth.
- Refspec (Maybe)
  - Required for *export*.
  - Optional for *import*.
  - Adds support for constraining references to private namespaces.
- Bidi-import (Maybe)
  - Modifies *import*, for potential efficiency improvements.
  - “The fast-import commands cat-blob and ls can be used by remote-helpers to retrieve information about blobs and trees that already exist in fast-import’s memory”.
- Export-marks (Maybe)
  - Modifiest *export*.
  - Git provides output of internal marks table.
  - Potential efficiency improvements.

Capabilities: Miscellaneous

- Import-marks (Maybe)
  - More information needed.
  - Intuitively one would this it's the reverse of *export-marks*, but not according to Git's docs.
- Signed-tags (Maybe)
  - Modifies *export* capability regarding how signed tags are passed to git-fast-export.
  - Looking into how Git handles tags in *fast-import* and *fast-export* is worthwhile.
- Object-format (Maybe)
  - Indicates the helper can use explicit hash algorithm extensions when interacting with remote.
  - Potentially useful for supporting alternative hash algorithms, as done by OCI.
- See Miscellaneous Capabilities

Questions?

Sources

- Git logos by Jason Long - CC BY 3.0
- Open Container Initiative logos obtained from GitHub - Apache 2.0
- Zarf logo - Apache 2.0
- Git remote helpers
- ORAS
- ASCE Data Tool