Working Session

Feb 2

Meeting Agenda

- Configuration File Spec
- Web Service Spec
- Branches and Merging
- DevOps Consultant

Configuration File Spec

- The specification is in the LOST requirements/design document
- Please try to implement yourself from the spec
- Example code for reading the configuration file is in the instructor's LOST repo

The configuration file format has been documented in the LOST spec document in the Configuration section.
The code is not very hard to write but does take a little googling and documentation reading to figure out what to call and how to figure out where the application is running. Please try to implement this on your own.
That said, the configuration file reader doesn't have much logic in it, so I expect most solutions will look a lot a like.

Web Service Spec

- Next sprint - web service API implementation
- Spec has been written but I would like to provide some test code that includes the crypto so that you have an example.

Web Service Spec

- 6 Webservice API Calls
  - lost_key(arguments, signature)
  - activate_user(arguments, signature)
  - suspend_user(arguments, signature)
  - list_products(arguments, signature)
  - add_products(arguments, signature)
  - add_asset(arguments, signature)
- ...and an HTML page that describes how to use these; copy as much as you like from the spec...

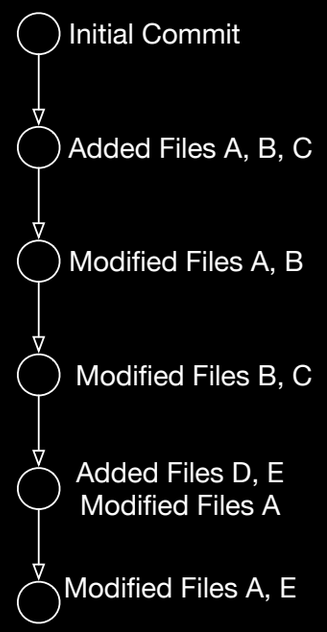
Arguments is contains JSON encrypted using the LOST public key. Signature is the cryptographic signature of the encrypted JSON. The signature is generated using the sending application's private key.
The JSON for each of the requests and responses is given in the document. Read the document and ask questions if things are unclear.

I would approach this sprint by first generating the paths and templates I think I would need. Then I would implement the API using plaintext, making it much easier to debug. Last I would introduce the encryption code.

The encryption for me is the highest risk component, that's why I'm starting by making some test code for you… When I'm not coding, I would be thinking about challenges and structure to support the encryption requirement.

Branches, Merges, Tags

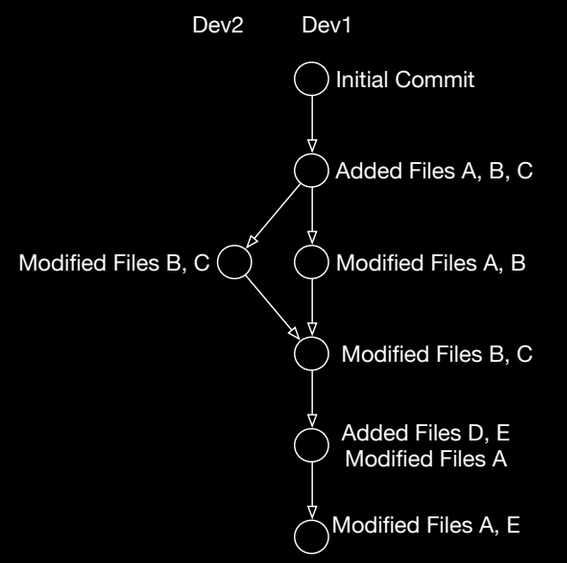
- Limited conflict as a single developer
- Developer as the only user is always aware of code stability
- Life is simple:
  - One branch
  - No merging
  - No tags



As a single developer/user version control is pretty easy. Few things happen concurrently.

Branches, Merges, Tags

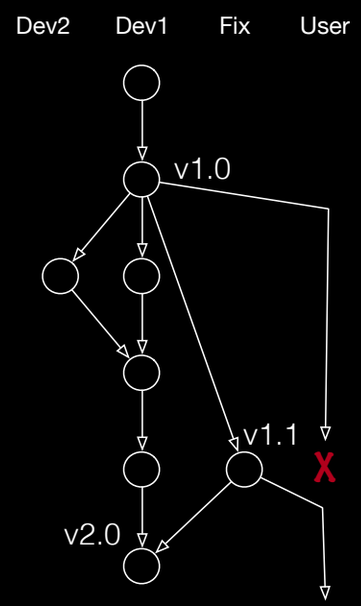
- Two developers mean trouble
- Conflicting changes
  - Overwrite?
  - Merge?
- A little more trouble:
  - Branches
  - Simple merges?
  - No tags



With multiple developers, concurrent changes can result in conflicting changes. These changes may require manually merging code.

Branches, Merges, Tags

- Releases are a challenge...
- Which code do I debug?
- A little more trouble:
  - Branches
  - Merges
  - Probably Tags



When non-developer are added there are some additional problems. The code under development is not sufficiently stable for the user, so it can’t just be rolled out. When defects are detected in the customer’s version, changes generally need to be made from the commit at which the released version was cut.