Documentation checklist

The purpose of this document is to provide a framework for analyzing your project and infrastructure in order to capture all revelant documentation. Some sections may not apply in your case. Or, there may be something very unique about your environment that is not covered here. Use this as a guide and a starting point.

Documentation is only useful if it can be easily found, and if it is kept up to date. High-level documentation should exist in one central spot, not tucked away inside one particular project or repository. One useful approach is to create a dedicated repo called `infrastructure` in GitHub, and use it for those high-level docs. If you chose to use the Wiki functionality in Github, include a prominent link in the repo's `README` explaining that is where the documentation can be found.

> *Even if you only have one project/repo currently, it's still a good idea to keep the high-level documentation in a separate, central place. That way, when you add a second repo, you don't have to shuffle things around.*

Keeping documentation up to date requires process and discipline. Speak with the team and any stakeholders about the need to keep documentation up to date. If anyone sets up a new service, changes service configuration, or decommissions a service, they should also take responsibility to include documentation. One benefit of using Github for documentation is that you can use your normal code review process to make sure documentation is accurate and up to date.

Action items

- ☐ Decide on a location for high-level documentation. Create an `infrastructure` repo, if necessary.
- ☐ Communicate with the team and stakeholders about how to find and contribute to documentation.
- ☐ If you have a code review process or checklist, add documentation review to that process.

Credentials

This document will refer to many different services that require login credentials. Do not add the credentials to your documentation. You should be able to freely share this documentation with your team. If you add credentials directly to the document, you would only be able to share it with those that should have full access to every documented service.

Instead, you should use this section of the document to explain where credentials can be found. Then, you can use other access control mechanisms to decide how to share credentials when necessary. For example, popular password managers have team functions or shared vaults, where you can invite specific individuals to have access to specific credentials.

Credential sharing should be considered a last resort, or as a disaster recovery scenario. Most services have functionality to manage multiple users on a team. Instead of sharing one common account, leverage those user management features. It may require you to pay for a higher tier of service, but this is usually worth the added security and functionality.

Consider AWS as an example: No one should ever need to share AWS credentials for an account. See our AWS Account Setup guide for instructions on how to invite other developers, even contractors outside your organization, to your AWS account. That particular setup guide assumes inviting a trusted developer who can manage everything, but you could alter the Role permissions to allow access to only specific services.

That being said, there is one root AWS account which has certain access which cannot be fully delegated. It is critical to make sure this root AWS account is shared with more than one person in your organization. Losing access to that can be extremely difficult to recover from. This is the disaster recovery scenario mentioned above.

The other scenario where credential sharing may be necessary are for secrets like API keys or for services that don't have any sort of user management capability. In these cases, there is no other choice but to share a single credential, but this should represent a very small percentage of your credentials.

It may require some research and team communication to identify all these credentials. As you start documenting it, you may identify credentials managed by a team member or accounts owned by a third party (perhaps an outside contractor).

Action items

- [ ] Set up a mechanism for sharing credentials (for example, 1Password).
- [ ] Capture all existing credentials into that central secure store.
- [ ] Interview the team to identify "one-off" services that were set up without your knowledge.
- [ ] Migrate "one-off" or third-party-owned services to be under your control.
- [ ] Invite at least one highly-trusted person to share all credentials for disaster recovery purposes.
- [ ] Invite team members to share credentials for secrets or services without user management.
- [ ] Review all previous use of shared credentials to identify services with user management features and migrate to user accounts.
- [ ] Reset any passwords previously shared that are now migrated to individual user accounts.
- [ ] Create a Credentials section in your high-level documentation.
- [ ] Document the mechanism for sharing credentials.
- [ ] Document the primary and disaster recovery contacts that will manage shared credentials.