Git creation and editing

Issues

- JENKINS-43148 - Getting issue details... STATUS

Problems

- The largest notable difference is that creating and editing with plain Git has noticeably more network traffic than Github. This means the user may have to wait for Blue Ocean to retrieve the content of the repository before showing them the Editor or before their save action is completed.
- The administrator is delegating the control of their credentials to read/write to the repository. They may want finer grained controls for who can edit the Jenkinsfile.

Assumptions and limitations

- We will use whatever credential was provided in the first instance to setup the Pipeline to write back to the Git repository.
- If writing fails, we let them know the key they provided isn't allowed to push and the user is given instructions on how to copy the Jenkinsfile to their repository.
- All other situations do not matter.
- User will still be able to setup a system SSH key on the server and allow users to write with it desired but only if they picked system credential when they initially setup the Pipeline.
- We are also assuming that the URL to the repository is the same URL that will be used for the write. This means there is no option to enter a different remote URL on write or any credential negotiation that goes with that flow.

Security scenario

In Github read/write

If Bob sets up the Pipeline he delegates his Github token for reading from the repository, for the purpose of running the Pipeline. When Alice comes along and wants to edit the Pipeline, she cannot use Bobs access token and is prompted to provide her own access token used for editing the pipeline.

Note that today, in order for Alice to edit, she needs to go and enter her token in creation then click the edit button and then she can author. This needs to be fixed properly as part of JENKINS-42791 - Getting issue details... STATUS .

In Git read/write

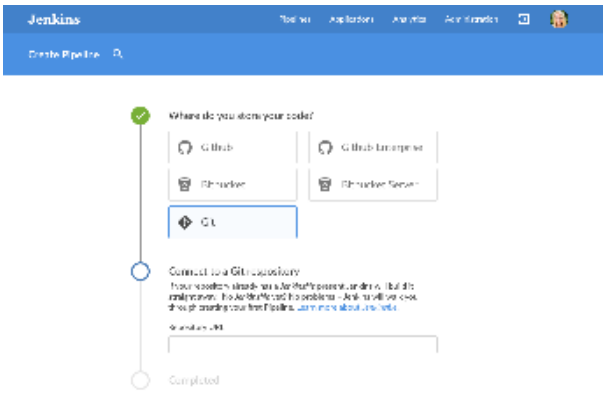
SSH credentials

If Bob sets up the Pipeline he uses his Jenkins public key to delegate his Git ssh key for reading from the repository, for the purpose of running the Pipeline. When Alice comes along and wants to edit the Pipeline, she cannot use Bobs ssh key. Jenkins generates a public/private key pair and stores it as a credential against her user. She is then prompted to download a public key and register it with the Git server.

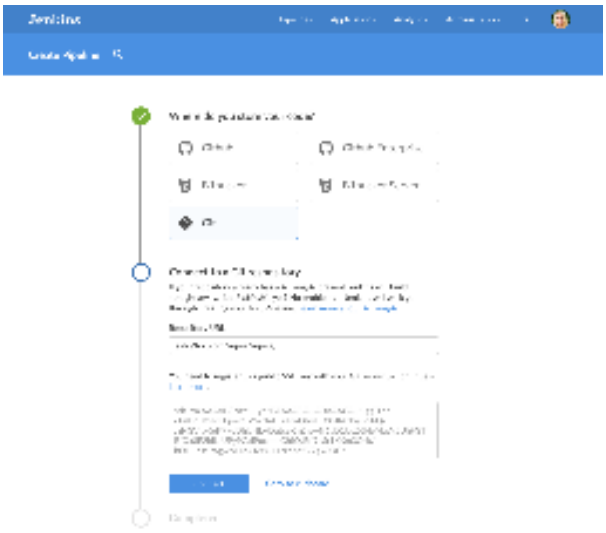
User flows

1. First step

Developer enters in their URL and based off of the protocol we decide if they will use SSH key (SSH) or Username/password (http/https)



Creating the Pipeline using SSH



Loading the editor:

1. Developer clicks the edit action
2. Developer sees a progress dialog with a message "Loading your Jenkinsfile"
    - This could take a while as we have to do a shallow clone of the repository
    - How much progress information do we get from the clone? If we can easily get this info we can use a determinate progress indicator rather than a indeterminate one.
    - Developer should be able to cancel the load if it takes too long
3. Developer sees the Editor

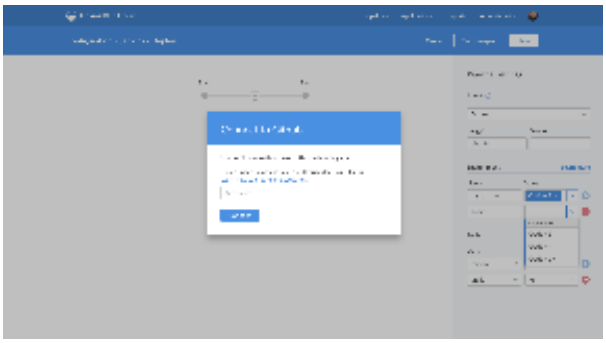
Saving from the editor

1. Developer clicks save
2. Developer sees the save dialog and then confirms
    a. May commit back to new branch or current branch
3. Developer sees progress dialog with message "Saving your pipeline"
4. Developer lands back on the Activity screen

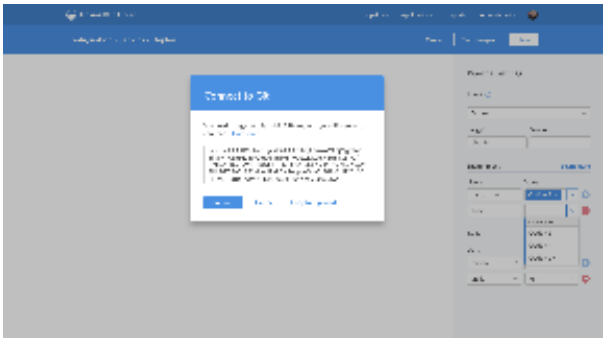
Whenever credentials are required

1. Blue Ocean detects the type of credential needed based on the repository URL
    - If Bob sets up the Pipeline using a Git URL with the SSH protocol then he can only create a SSH credential for it.
    - If Alice edits the Pipeline that Bob setup using SSH then she needs a SSH key
2. User is presented with a way of creating their own credential (as scoped in the Security scenarios) and credential is validated before allowing the user to continue
3. User can perform creation or editing actions

Github with access token



Git with Jenkins user public key



Technical questions and problems

- Is there a smart repo cache that we can use on the master to perform the editing operations on?
- Is there any way to get progress information from the Git operations we use? We would like to display them to the user.
- To inform design – when asking for credentials and their remote URL, what other information could they need to successfully push the Jenkinsfike?

Keith's technical brain dump

I have this working by:

1. cloning the repo (shallow)
2. checkout the branch
3. reading the content

and for writes:

1. cloning the repo (shallow)
2. Depending if the branch has changed:
    a. creating a branch based on the source
3. saving content
4. adding a remote for the writable URL & credential
5. committing and pushing content