

Git: handling files

File created

You create a new
file inside a git
repo

file.py

This file is detected
but ignored by git
unless you
add

File created

You create a new
file inside a git
repo

file.py

This file is detected
but ignored by git
unless you
add

File added

Git is now
monitoring if the
file is changed.
If you change it further, just
add it again.

file.py

When you're
happy, you add
the file to the
staging area by
doing a **commit**

File created

You create a new file inside a git repo

file.py

This file is detected but ignored by git unless you **add**

File added

Git is now monitoring if the file is changed.
If you change it further, just **add** it again.

file.py

When you're happy, you add the file to the staging area by doing a **commit**

File committed

Commits advance the status of your DAG.

file.py

A commit creates a set of files ready to **push**. Pushing is just sending all your commits to the remote hosting site.

File created

You create a new file inside a git repo

file.py

This file is detected but ignored by git unless you **add**

File added

Git is now monitoring if the file is changed.
If you change it further, just **add** it again.

file.py

When you're happy, you add the file to the staging area by doing a **commit**

File committed

Commits advance the status of your DAG.

file.py

A commit creates a set of files ready to **push**. Pushing is just sending all your commits to the remote hosting site.

File pushed

The changes now reside in your remote repo.

file.py

If someone in your team wants to access the changes you made, they need to **pull** them.

File created

You create a new file inside a git repo

file.py

This file is detected but ignored by git unless you **add**

File added

Git is now monitoring if the file is changed.
If you change it further, just **add** it again.

file.py

When you're happy, you add the file to the staging area by doing a **commit**

File committed

Commits advance the status of your DAG.

file.py

A commit creates a set of files ready to **push**. Pushing is just sending all your commits to the remote hosting site.

File pushed

The changes now reside in your remote repo.

file.py

If someone in your team wants to access the changes you made, they need to **pull** them.

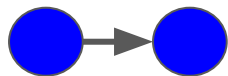
pull

Branches

Initial
commit



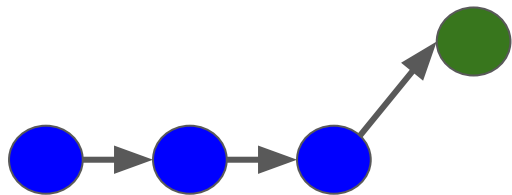
Some
changes

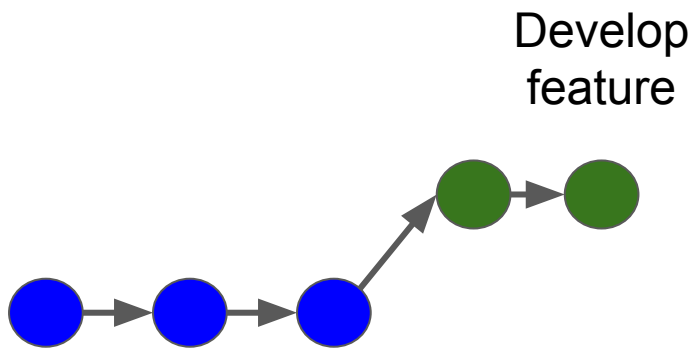


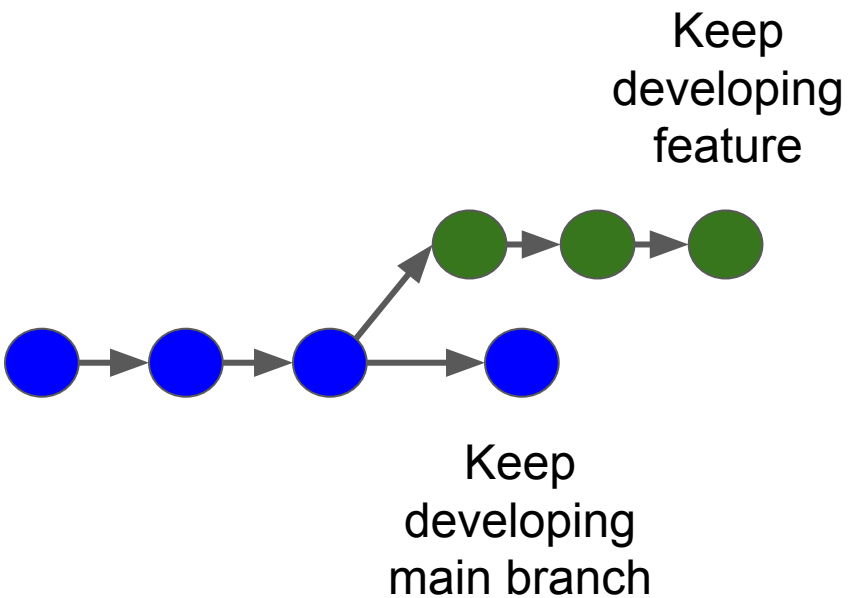
Working
stage

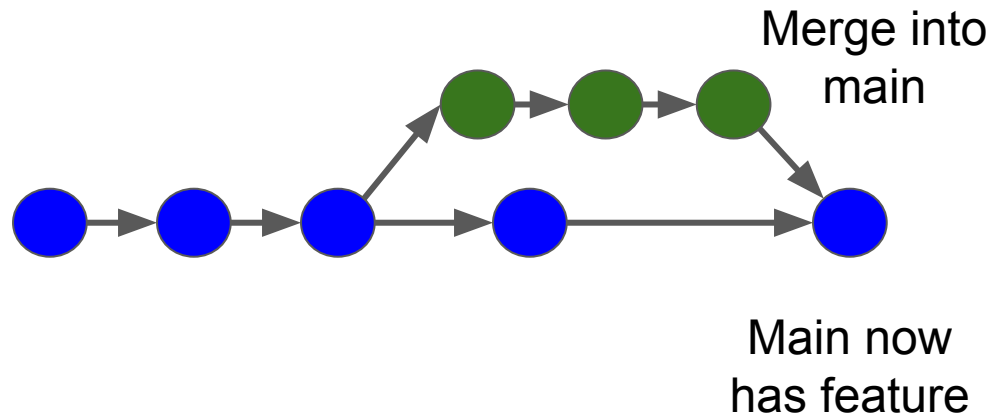


Feature
branch

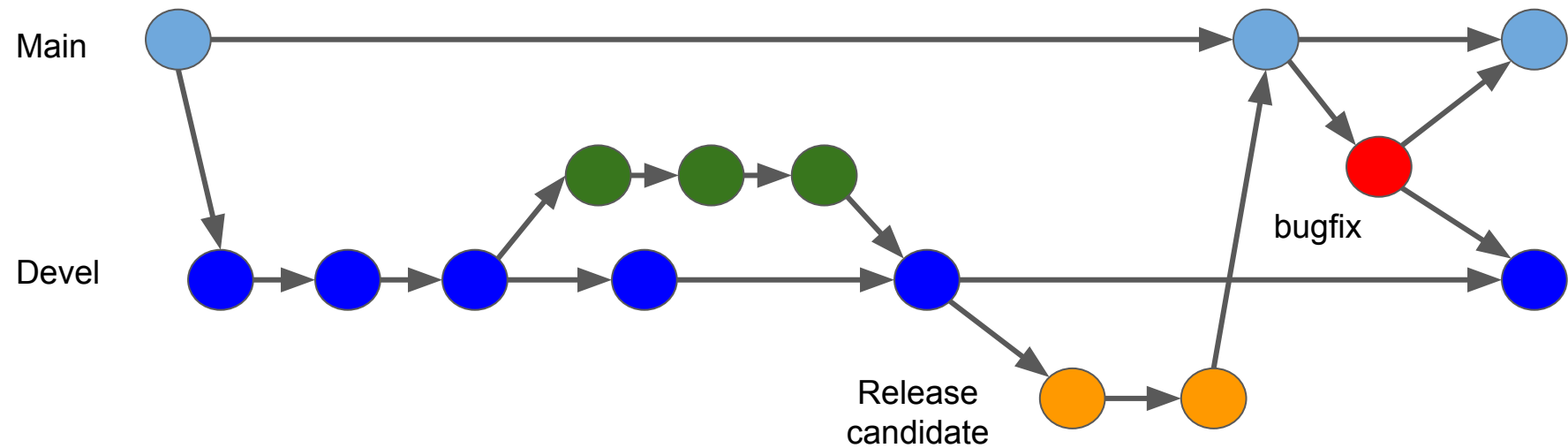








Complex development DAG



[SourceTree: free git helper](#)

[Auxiliary tool to manage git flows](#)

Working rationale



Local repository

The diagram consists of two rectangular boxes. The top box is light gray and contains the text 'Local repository'. The bottom box is blue and contains the text 'Working directory'. To the right of each box is a descriptive text label. The boxes are vertically aligned, with the gray box above the blue box.

Local version
of the git
repository

Working directory

Directory where you
are working on your
project



A diagram showing three components of a Git workflow arranged vertically. At the top is a light gray rectangle labeled 'Local repository'. In the middle is a blue rounded rectangle labeled 'Staging area', with a text block to its right explaining its purpose. At the bottom is a blue rectangle labeled 'Working directory'.

Local repository

Staging area

Area where you
prepare your code to
be committed

Working directory



A diagram illustrating the components of a Git repository. It consists of four rectangular boxes arranged in two columns. The left column contains a light gray box at the top labeled 'Local repository', a blue rounded rectangle in the middle labeled 'Staging area', and a green rounded rectangle at the bottom labeled 'Stash'. The right column contains a light blue box at the top labeled 'Working directory', a blue rounded rectangle in the middle labeled 'Staging area', and a blue box at the bottom labeled 'Working directory'. The boxes are connected by lines, indicating the flow of data between them.

Local repository

Auxiliary location:
Where you save your
current progress if you
want to return to the
beginning of the last
commit: we will not cover it
in the demo due to time
restrictions. You might
need to use it at some
point.

Stash

Staging area

Working directory

Local repository

Staging area

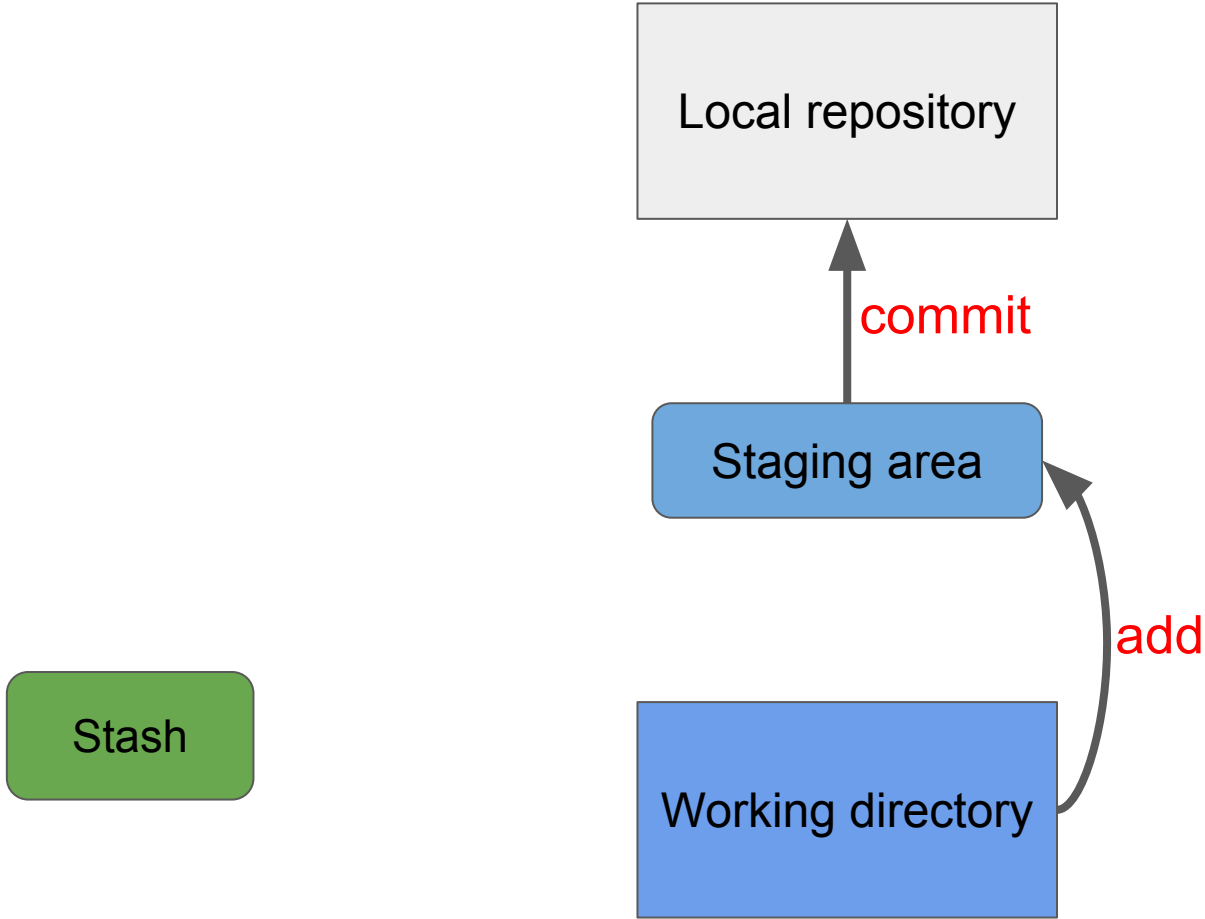
Stash

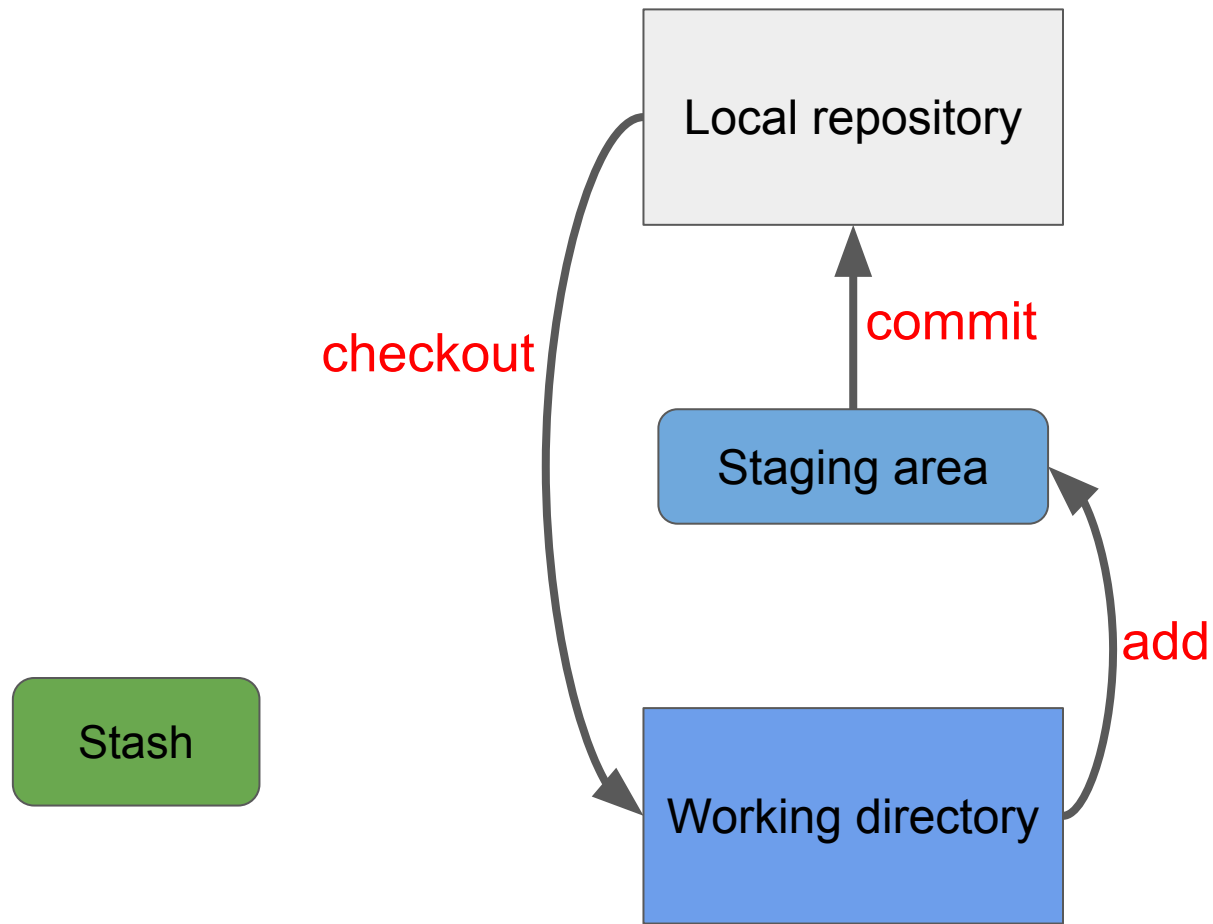
Working directory

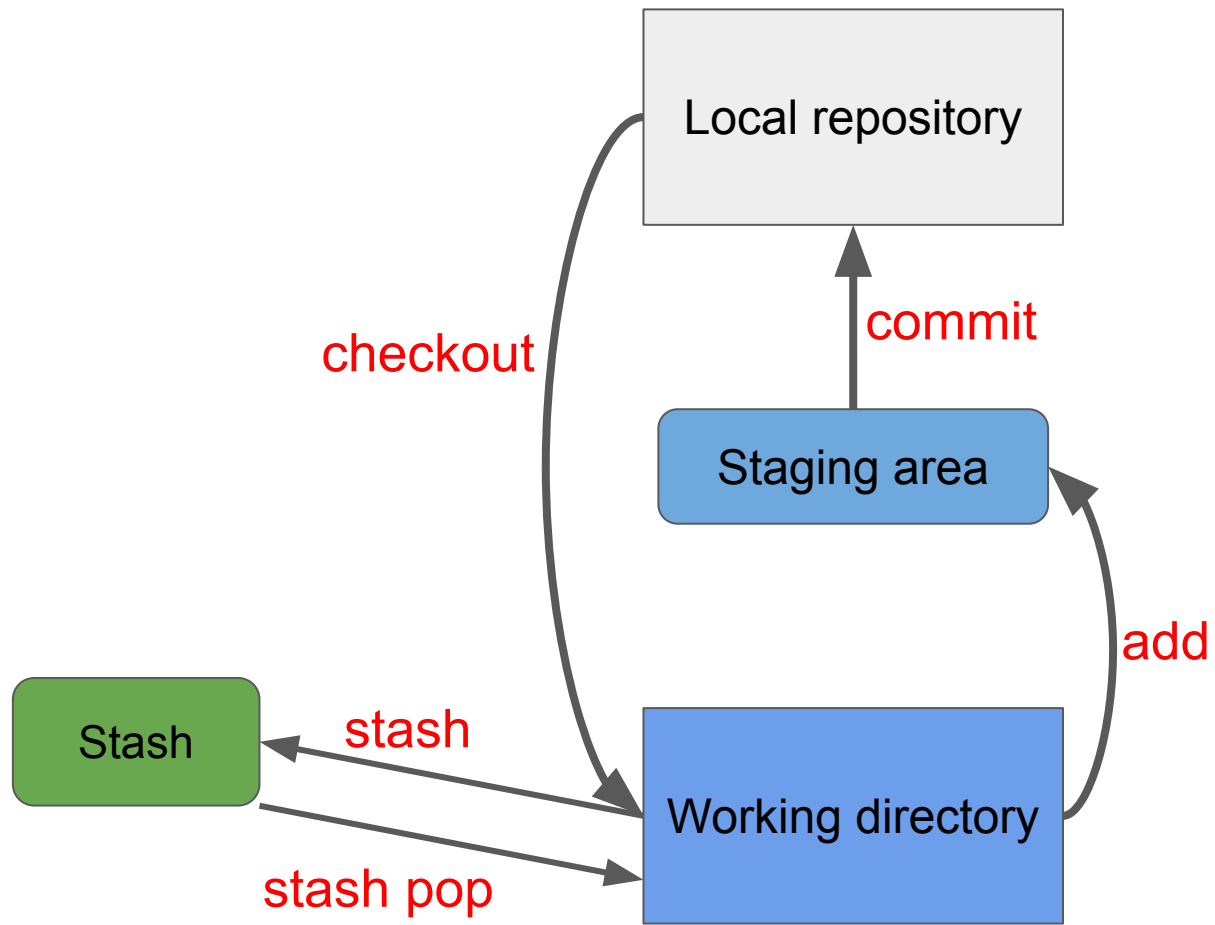
add

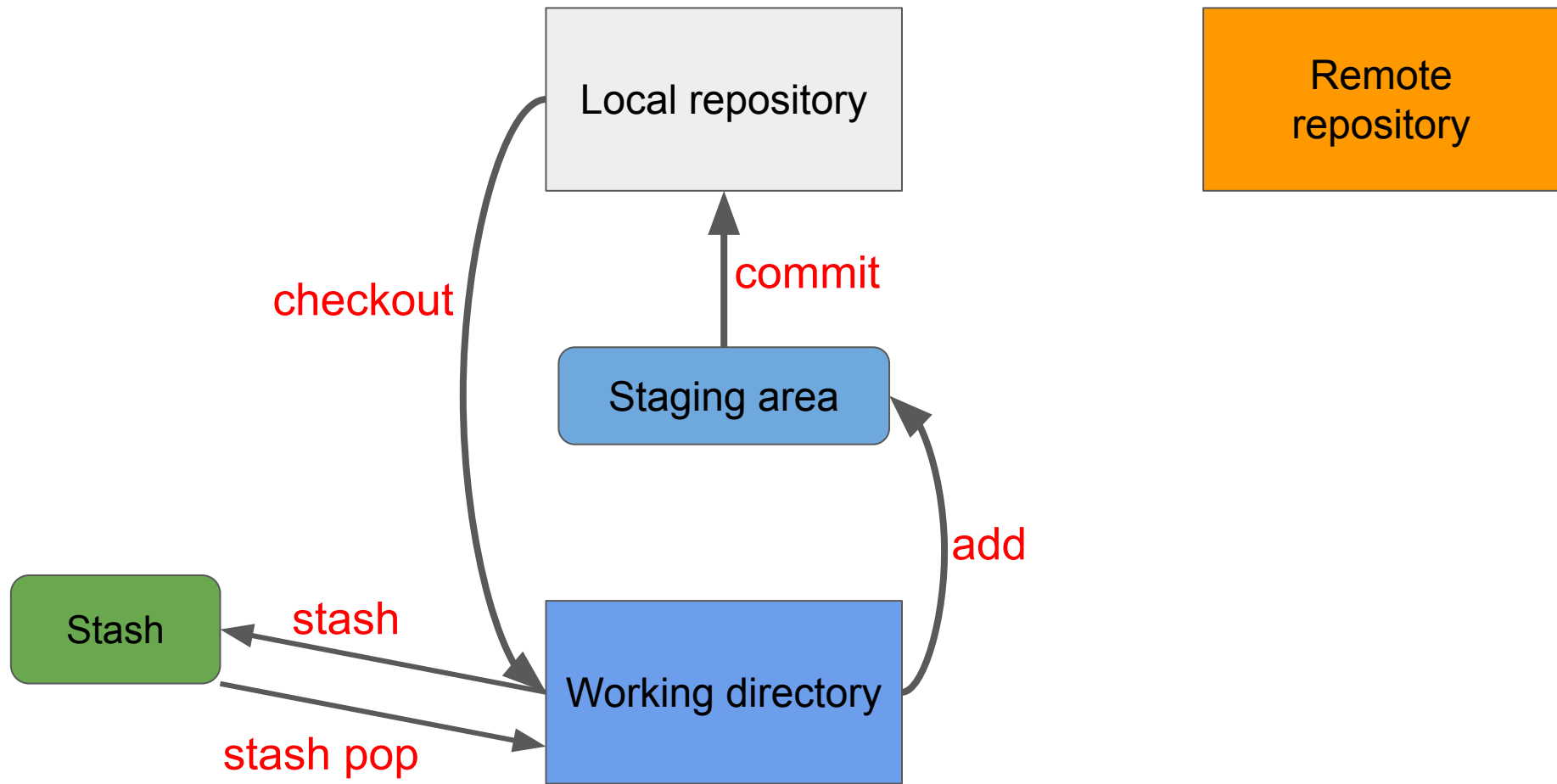
```
graph TD; LR[Local repository]; SA[Staging area]; WD[Working directory]; S[Stash]; WD -- add --> SA;
```

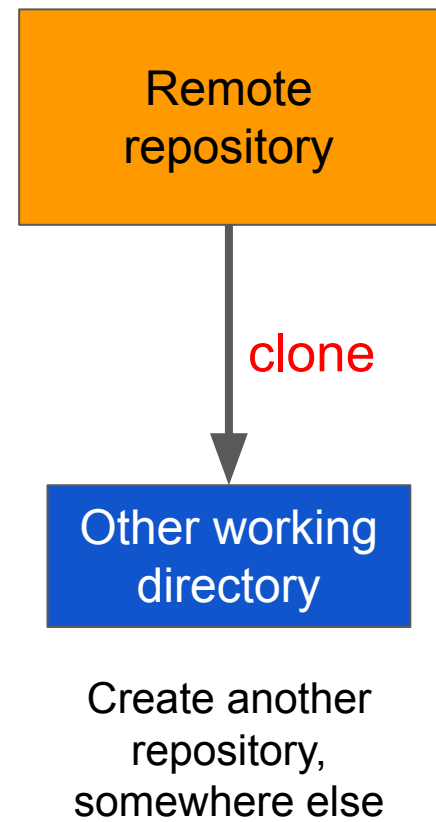
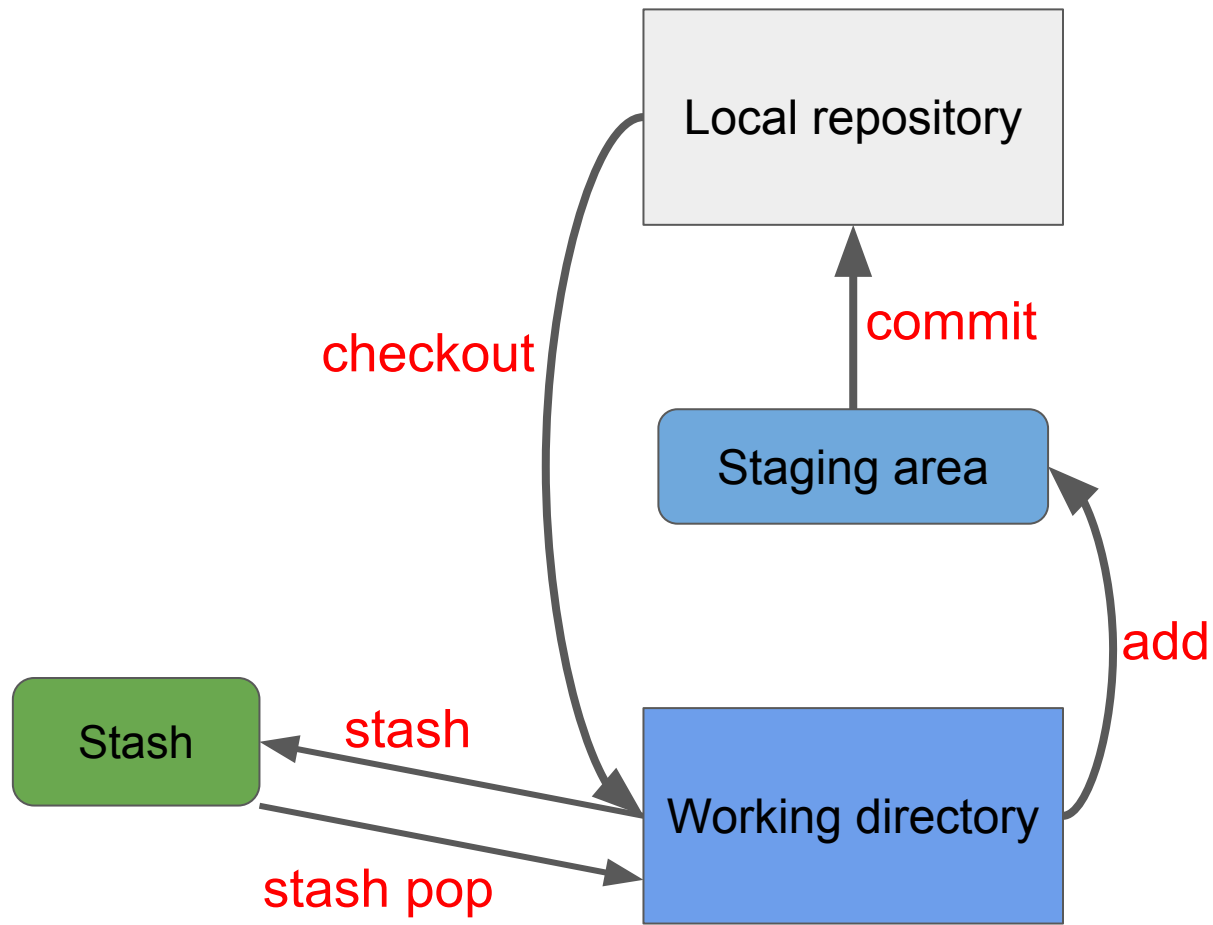
The diagram illustrates the components of a Git workflow. It features four main boxes: a light gray rectangular box at the top labeled 'Local repository'; a blue rounded rectangular box in the middle labeled 'Staging area'; a blue rectangular box at the bottom labeled 'Working directory'; and a green rounded rectangular box on the left labeled 'Stash'. A curved gray arrow points from the 'Working directory' box to the 'Staging area' box, with the word 'add' in red text positioned next to the arrow, indicating the command used to move changes from the working directory to the staging area.

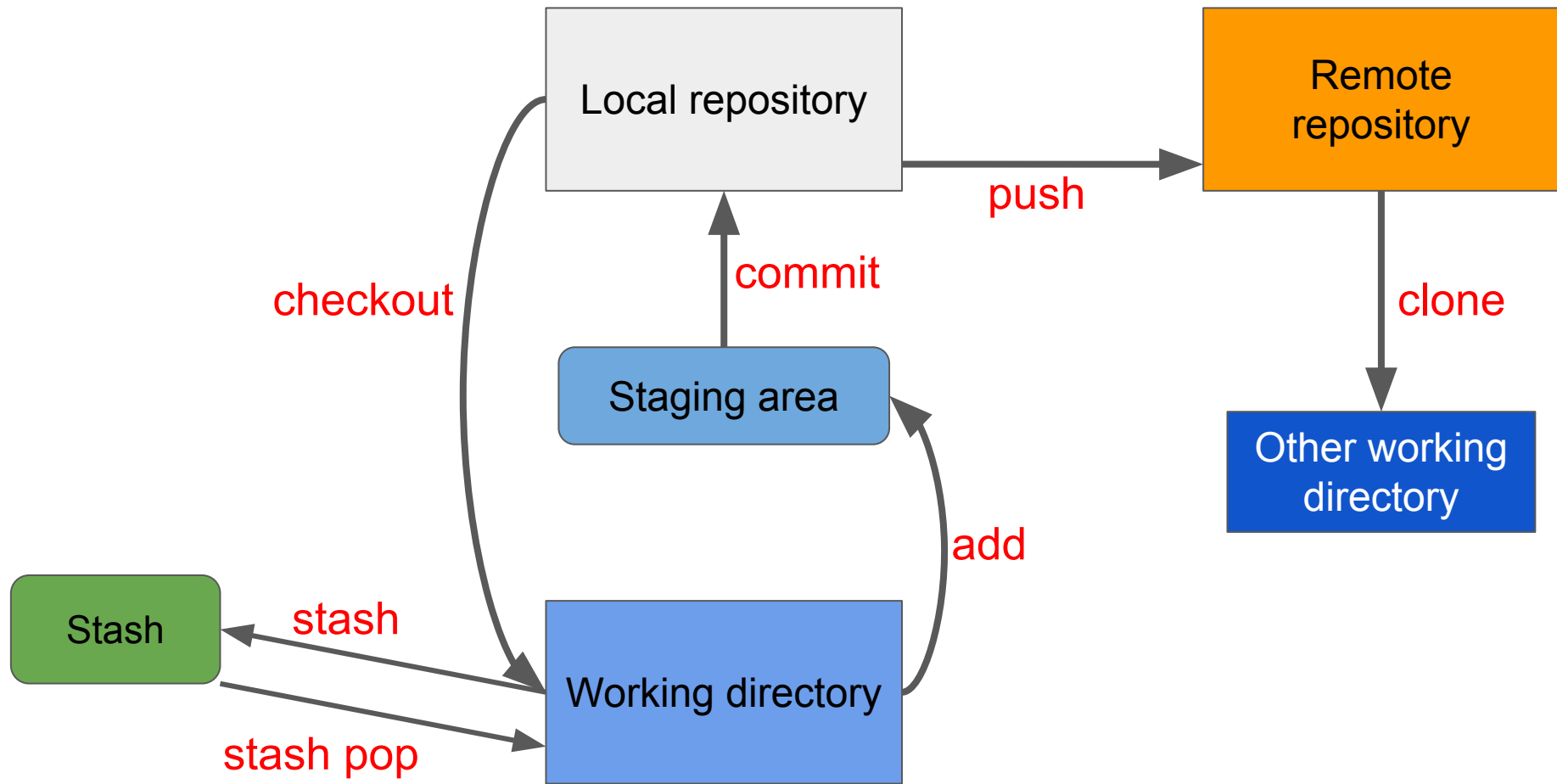






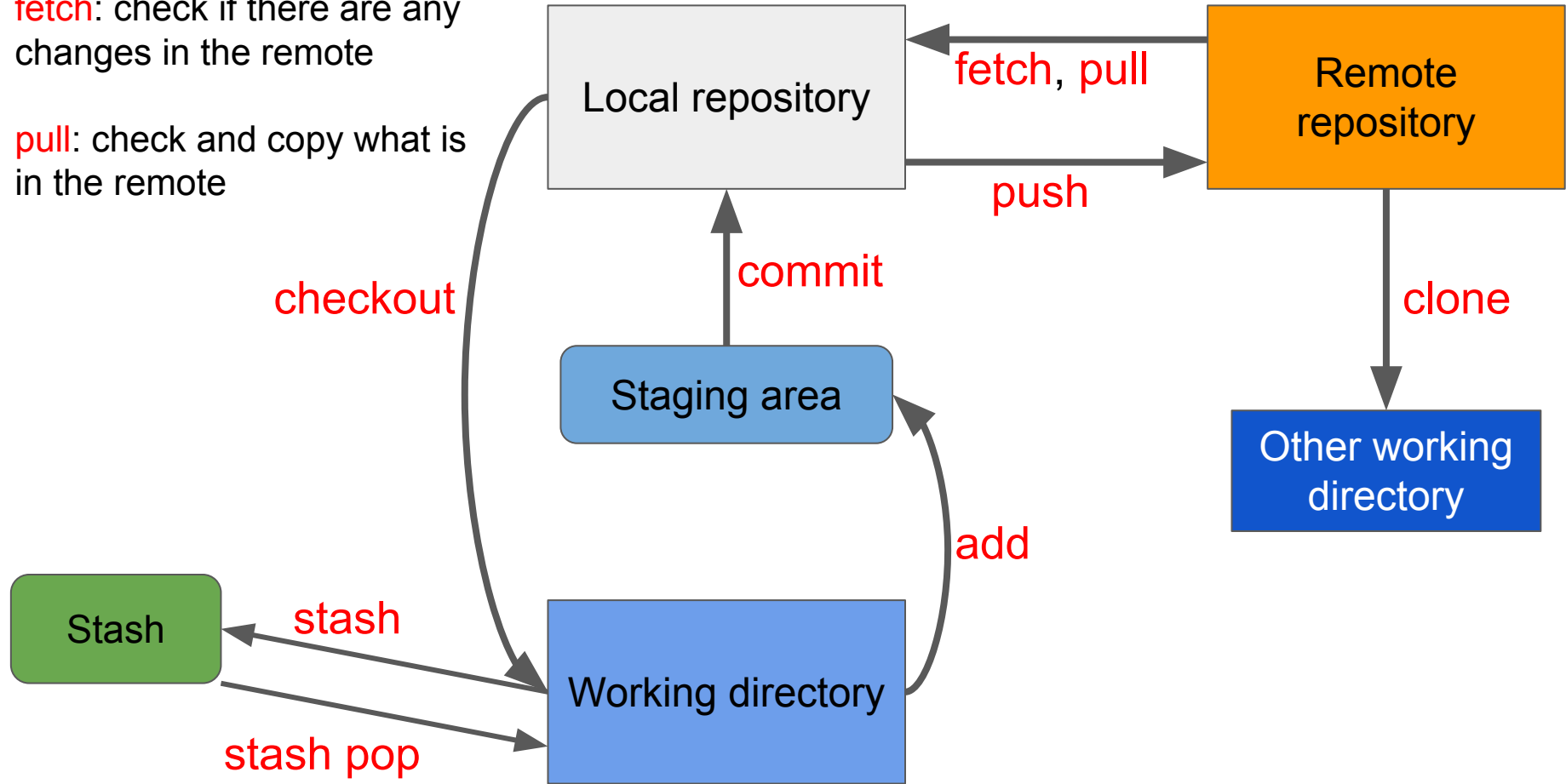






fetch: check if there are any changes in the remote

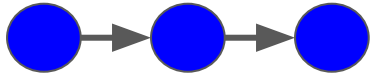
pull: check and copy what is in the remote



Demo images

Initial
commit

Working
stage



Some
changes

