What Builds With What?

The Development Branches

Swift	LLVM Project
main	swift/main

`main` is the place for active development on Swift. If you're just working on Swift, that's where you'll spend most of your time.

LLVM Project repo automatically merge changes from the latest release branch (see below) into `swift/main`. This generally means `swift/main` is just an alias for the latest release branch. If you want to do Swift-related development on LLVM projects, see "The Upstream Branches" below.

To switch from one set of branches to another, you can use `utils/update-checkout` in the Swift repository with the `--scheme` option. You can use any of the branch names as the argument to `--scheme`: in this case, either `main` or `swift/main`.

The Release Branches

Swift	LLVM Project
release/x.y	swift/release/x.y

At some point before a release, a *release branch* will be created in every repository with a name like `release/5.3`. (The actual number is chosen by Apple.) After the branch has been created, commits must make it to this branch to make it into the release. In some cases, the release manager for the branch will decide to merge in all additional changes from `main`; otherwise, cherry-picking changes and making a new pull request is the way to go. If there are any "patch" releases (e.g. Swift 5.3.1), they will also come from this branch.

Note that these branches come not from the "development" branches (above), but the "upstream" branches (below). This is because they need to contain the latest changes not just from Swift, but from the LLVM project as well. For some releases, the release branch for the LLVM project will be timed to coincide with the corresponding llvm.org release branch.

The Upstream Branches

Swift	LLVM Project
next	swift/next

`swift/next` is a branch for LLVM that includes all changes necessary to support Swift. Changes from llvm.org's main branch are automatically merged in. Why isn't this just `swift/main`? Well, because LLVM changes *very* rapidly, and that wouldn't be very stable. However, we do want to make sure the Swift stuff keeps working.

If you are making changes to LLVM to support Swift, you'll probably need to work on them in `swift/main` to test them against Swift itself, but they should be committed to `swift/next`, and cherry-picked to the current release branch (`swift/release/x.y`) if needed. Remember, the release branches are automerged into `swift/main` on a regular basis.

(If you're making changes to LLVM Project that *aren't* about Swift, they should generally be made on llvm.org instead, then cherry-picked to the active release branch or `swift/main`.)

`next` is an effort to keep Swift building with the latest LLVM changes. Ideally when LLVM changes, no Swift updates are needed, but that isn't always the case. In these situations, any adjustments can go into Swift's `next` branch. Changes from Swift's `main` are automatically merged into `next` as well.

Reference

Updating

```
swift/utils/update-checkout --scheme [branch]
```

You can use any of the branch names as the argument to `--scheme`, such as `main` or `swift/main`. See `update-checkout --help` for more options.

Committing

- Swift: new commits go to `main`
- LLVM Project: the destination branch depends on the kind of change that must be made:

    1. LLVM Project changes that don't depend on Swift
        - New commits go to `main` in the upstream llvm-project.
        - Then cherry-pick these commits to an appropriate, `swift/main` aligned `apple/stable/*` branch in Apple's fork of llvm-project. Please see Apple's branching scheme document to determine which `apple/stable/*` branch you should cherry-pick to.

    Note that **no new changes should be submitted directly to `apple/main`**. We are actively working on eliminating the differences from upstream LLVM.

    2. Changes that depend on Swift (this only applies to LLDB)
        - New commits go to `swift/main` (*not* an `apple/stable/*` branch, as these shouldn't contain changes that depend on Swift).
        - Then cherry-pick these commits to `swift/next`.
        - If necessary, cherry-pick to the release branch (`swift/release/x.y`), following the appropriate release process. (Usually this means filling out a standard template, finding someone to review your code if that hasn't already happened, and getting approval from that repo's *release manager.)*

    In the long term we want to eliminate the differences from upstream LLVM for these changes as well, but for now there is no concrete plan. So, submitting to `swift/next` continues to be allowed.

Automerging

Some branches are *automerged* into other branches, to keep them in sync. This is just a process that runs `git merge` at a regular interval. These are run by Apple and are either on a "frequent" (sub-hourly) or nightly schedule.

Swift

- `main` is automerged into `next`

LLVM Project

- `swift/release/x.y` (the *latest* release branch) is automerged into `swift/main`
- llvm.org's `main` is automerged into `swift/next`
- llvm.org's release branch *may* be automerged into `swift/release/x.y`, if they are in sync