rust-lang.org/tools/install

Rust

Install Learn Playground Tools Governance Community Blog

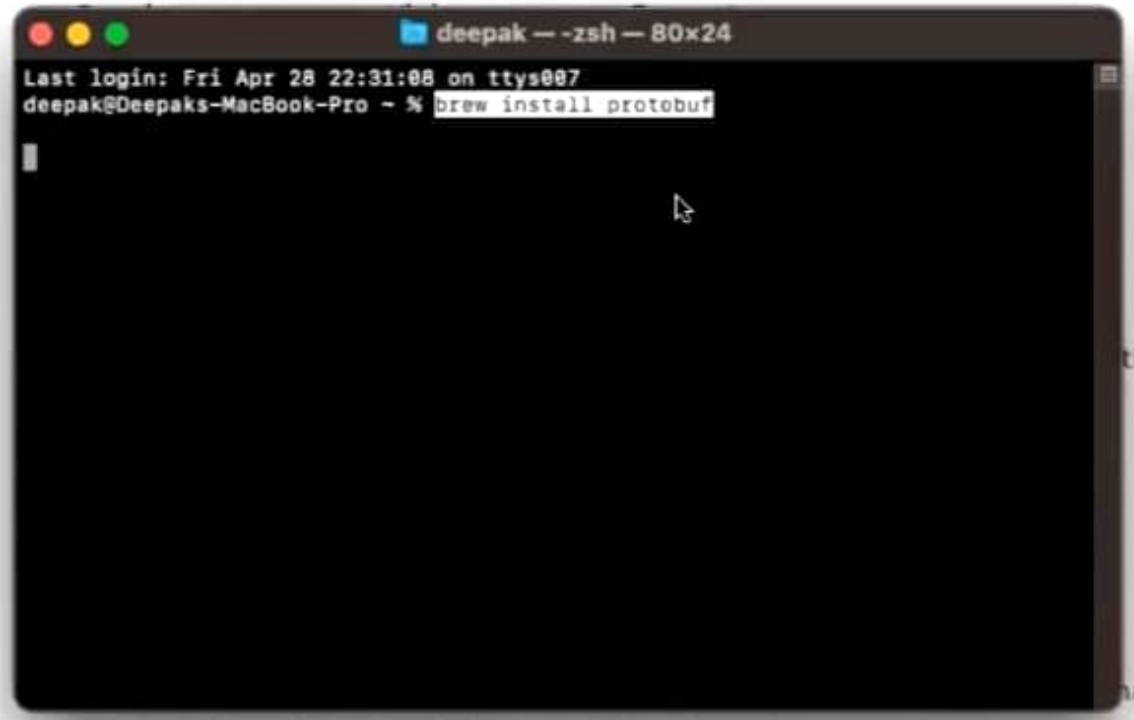
Install Rust

Using rustup (Recommended)

It looks like you're running macOS, Linux, or another Unix-like OS. To download Rustup and install Rust, run the following in your terminal, then follow the on-screen instructions. See "Other Installation Methods" if you are on Windows.

```
curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh
```

Notes about Rust installation



the

macOS computers. If you don't already have Homebrew installed on your local computer, you should download and install it before continuing.

To install Homebrew:

1. Open the Terminal application.
2. Download and install Homebrew by running the following command:

BASH

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.c
```

deepak — -zsh — 80×24

```
Default host: aarch64-apple-darwin
rustup home:  /Users/deepak/.rustup

installed toolchains
--------------------

stable-aarch64-apple-darwin
nightly-2022-06-30-aarch64-apple-darwin
nightly-aarch64-apple-darwin (default)

installed targets for active toolchain
--------------------------------------

aarch64-apple-darwin
wasm32-unknown-unknown

active toolchain
----------------

nightly-aarch64-apple-darwin (overridden by +toolchain on the command line)
rustc 1.68.0-nightly (1e4f90061 2023-01-11)

deepak@Deepaks-MacBook-Pro ~ %
```

following

The command displays output similar to the following:

BASH

```
# rustup show

active toolchain
----------------

stable-x86_64-apple-darwin (default)
rustc 1.61.0 (fe5b13d68 2022-05-18)
```

Setup Your Project

1. Make sure you already have Rust build environment setup on your computer:

   https://docs.substrate.io/install/

   If you don't... probably best to just watch along.

2. Clone the `substrate-node-template`

```
git clone https://github.com/substrate-developer-hub/substrate
```

3. Replace the contents of `substrate-node-template/pallets/template/` with the **Empty Pallet Template**.

4. Compile your project. Don't worry about warnings :)

```
cargo build -p pallet-template
```

If you got this far, then you are ready to move forward!

Empty Pallet Template

```
#![cfg_attr(not(feature = "std"), no_std)]
```

Substrate_session — -zsh — 80×24

```
stable-aarch64-apple-darwin
nightly-2022-06-30-aarch64-apple-darwin
nightly-aarch64-apple-darwin (default)

installed targets for active toolchain
--------------------------------------

aarch64-apple-darwin
wasm32-unknown-unknown

active toolchain
----------------

nightly-aarch64-apple-darwin (overridden by +toolchain on the command line)
rustc 1.68.0-nightly (1e4f90061 2023-01-11)

deepak@Deepaks-MacBook-Pro ~ % cd Downloads
deepak@Deepaks-MacBook-Pro Downloads % cd Substrate\ session
deepak@Deepaks-MacBook-Pro Substrate session % cd ..
deepak@Deepaks-MacBook-Pro Downloads % cd Substrate_session
deepak@Deepaks-MacBook-Pro Substrate_session %
```

```
#[pallet::config]
pub trait Config: frame_system::Config {
    type RuntimeEvent: From<Event<Self>> + IsType<<Self as fr
}
```

Setup

1. Make sure

https://doc

If you don't

2. Clone the

```
git clone
```

3. Replace the

with the En

4. Compile yo

```
cargo bu
```

If you got this f

Welcome

Project

substrate-node-template ~/Downl

- .devcontainer
- .github
- .vscode
- docs
- node
- pallets
  - template
    - src
      - benchmarking.rs
      - lib.rs
      - mock.rs
      - tests.rs
    - Cargo.toml
    - README.md
- runtime
- scripts
- .editorconfig
- .gitignore
- Cargo.lock
- Cargo.toml
- CODEOWNERS
- LICENSE
- README.md
- rustfmt.toml
- shell.nix

External Libraries

Scratches and Consoles

README.md | template/.../lib.rs

No Cargo projects found — Attach — Do not show again

```rust
#![cfg_attr(not(feature = "std"), no_std)]

/// Edit this file to define custom logic or remove it if it is not needed.
/// Learn more about FRAME and the core library of Substrate FRAME pallets:
/// <https://docs.substrate.io/reference/frame-pallets/>
pub use pallet::*;

#[cfg(test)]
mod mock;

#[cfg(test)]
mod tests;

#[cfg(feature = "runtime-benchmarks")]
mod benchmarking;

#[frame_support::pallet]
pub mod pallet {
    use ...;

    #[pallet::pallet]
    pub struct Pallet<T>(_);

    /// Configure the pallet by specifying the parameters and types on which it depe
    #[pallet::config]
    pub trait Config: frame_system::Config {
        /// Because this pallet emits events, it depends on the runtime's definition
```

mock

Git | TODO | Problems | Terminal | Services | Build

Using rustup (moments ago) — Downloading crates ... — 9:10 LF UTF-8 Tab* main

substrate-node-template | Current File | Git:

README.md | template/.../lib.rs

No Cargo projects found — Attach — Do not show again

```rust
/// Edit this file to define custom logic or remove it if it
/// Learn more about FRAME and the core library of Substrate FRAME pallets:
/// <https://docs.substrate.io/reference/frame-pallets/>
pub use pallet::*;

#[cfg(test)]
mod mock;

#[cfg(test)]
mod tests;

#[cfg(feature = "runtime-benchmarks")]
mod benchmarking;

#[frame_support::pallet]
pub mod pallet {
    use ...;

    #[pallet::pallet]
    pub struct Pallet<T>(_);

    /// Configure the pallet by specifying the parameters and types on which
    #[pallet::config]
    pub trait Config: frame_system::Config {
        /// Because this pallet emits events, it depends on the runtime's def
        type RuntimeEvent: From<Event<Self>> + IsType<<Self as frame_system::
}
```

Empty Pallet Template

```rust
#![cfg_attr(not(feature = "std"), no_std)]

pub use pallet::*;

#[frame_support::pallet]
pub mod pallet {
    use frame_support::pallet_prelude::*;
    use frame_system::pallet_prelude::*;

    // The struct on which we build all of our Pallet logic.
    #[pallet::pallet]
    pub struct Pallet<T>(_);

    /* Placeholder for defining custom types. */

    /* Placeholder for defining custom storage items. */

    // Your Pallet's configuration trait, representing custo
    #[pallet::config]
    pub trait Config: frame_system::Config {
        type RuntimeEvent: From<Event<Self>> + IsType<<Self
    }
```

What is Substrate?

Substrate is a framework and toolkit to develop application specific blockchains. Application logic is encapsulated by writing specialized runtimes, which really is the executable that nodes use to run the blockchain logic of the network.

Substrate is built in Rust and uses WebAssembly for some key defining features. WebAssembly compatibility enables provability, verification and upgradability of runtimes, useful for on-chain governance and multi-chain consensus, as well as ensuring that blockchains built with it can be platform agnostic.

Substrate provides a way to easily write runtime logic using pallets written with FRAME — Substrate's opinionated toolkit for writing runtime logic. You can think of comparing the runtime of a Substrate blockchain to a crate, carrying all of its business logic in a multitude of different pallets.

In this workshop, we'll focus on writing a pallet for a blockchain specialized in managing the decentralized creation and ownership of crypto Kitties.

FRAME

The runtime of a Substrate blockchain acts as the state transition function, where all of the business logic of our chain lives. A runtime typically contains a number of pallets that, together, define the business logic of the chain. Each pallet alone provides ways for accounts and other pallets to interact with them. Pallets are written in FRAME, Substrate's opinionated framework for writing runtimes. Two key libraries that make it possible to write a pallet using FRAME are:

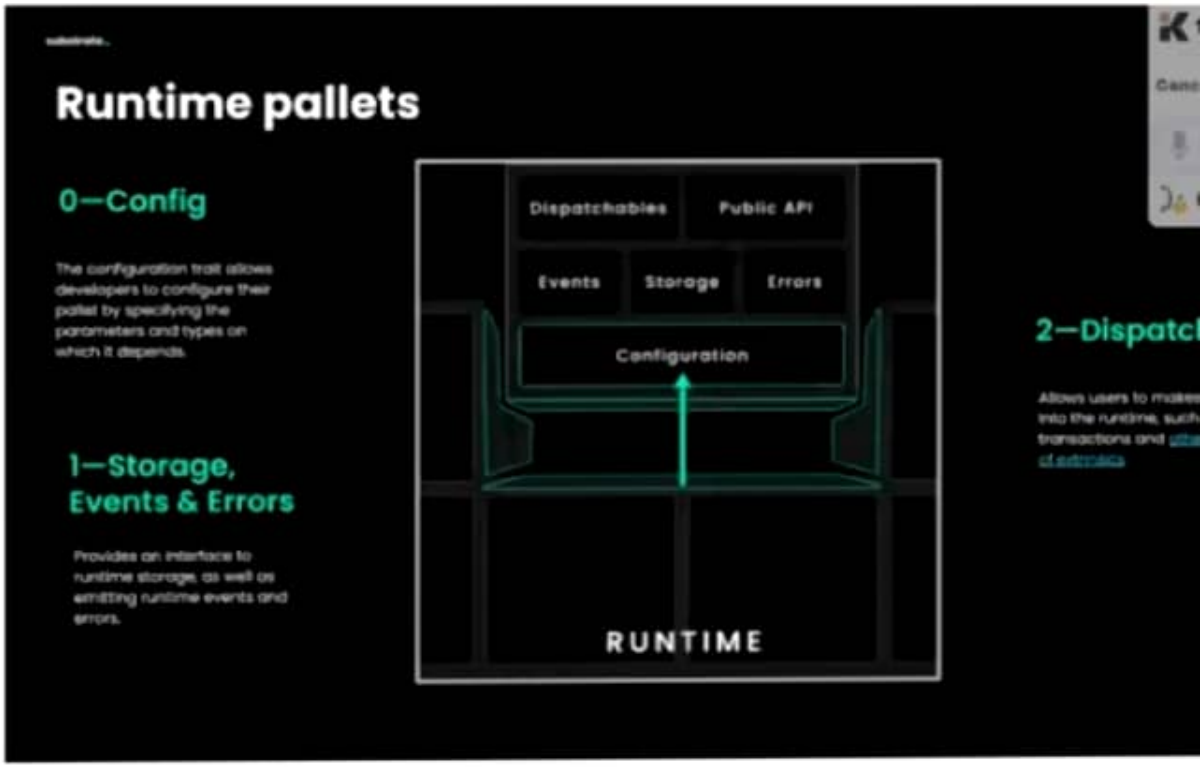
- `frame_system` : provides all core system functionalities for integrating a pallet into a runtime and enabling pallets to interact with each other.
- `frame_support` : provides a pallet with a way to create dispatchable calls, storage capabilities, events and errors and core utilities.

As runtime engineers, a pallet provides you a way to:

- Specify storage items for your blockchain
- Specify some callable functions for your blockchain
- Emit events and errors
- Specify some custom logic for callable functions
- Generate metadata

This gives you a lot of flexibility for the logic you're writing, but this means you have a lot of responsibility for carefully handling errors.

Previous
Substrate



Important concepts:

- Weights
- Verify first, write last
- Storage read/write efficiency
- Error handling

Configuring Your Pallet

To build our pallet, we need to include some custom configurations which will allow our pallet to gain access to outside interfaces like

- Manipulating user balances
- Generating our chain randomness
- Setting limits for how many kitties an single user can own

We will introduce these to the runtime configuration for our Pallet

To do this, this we use a few different tools

- Currency: A trait that describes an interface to access and manipulate user balances. Also gives you access to the AccountId type
- Randomness: A trait which simply fetches a random value, allowing the user to configure the Randomness source
- KittyIndex: A trait which describes an interface to access an on chain random value

We will use these interfaces in the future, but a sneak peak to how you might actually see these used in the code

```
type KittyIndex(KittyIndexFeature = u32 = u32);

type Currency = Balances;

#[frame_support::pallet]
pub mod pallet {
    use frame_support::{traits::{Currency, Ran
    use frame_system::{pallet_prelude::*};

    use frame_support::{traits::{Currency, Ra

    #[pallet::pallet]
```

SOLUTION

This should compile successfully by running:

```
cargo build -p pallet-template
```

Don't worry about warnings