



# Monitor Rust with Prometheus + Grafana

Application Monitoring



Grafana



# Create a file named main.rs

First, create a Rust application using the Rocket framework with a file named main.rs

```
***

#[macro_use] extern crate rocket;

use prometheus::{Registry, TextEncoder, Registry};
use rocket::response::content::RawHtml;
use rocket::serde::json::Json;
use serde::Serialize;
use std::sync::Arc;
use std::sync::Mutex;
use metrics::AppMetrics;

// In our model metrics
mod metrics;

#[derive(Serialize)]
struct JsonResponse {
    message: String,
    code: u16,
}

#[launch]
fn rocket() -> _ {
    let registry = Registry::new();
    let metrics = AppMetrics::new(&registry);

    let metrics = Arc::new(Mutex::new(metrics));

    rocket::build()
        .manage((metrics, registry))
        .mount("/", routes![index, routes![login, routes![read,
        crud_routes::update, crud_routes::delete]]
        .mount("/", routes![login, metrics_endpoint])
        .configure(rocket::Config {
            port: 8070,
            ..default()
        })
    )

    #[get("/")]
    fn index() -> Result<str> {
        "Hello, Bayu Wida Santoso!"
    }

    #[get("/metrics")]
    fn metrics_endpoint(metrics: Arc<Mutex<State<AppMetrics>>, Registry):
    -> Result<String> {
        let (_, registry) = metrics.lock().unwrap();
        let encoder = TextEncoder::new();
        let metric_families = registry.gather();
        let mut buffer = Vec::new();
        encoder.encode(metric_families, &mut buffer).unwrap();
        RawHtml(String::from_utf8(buffer).unwrap())
    }

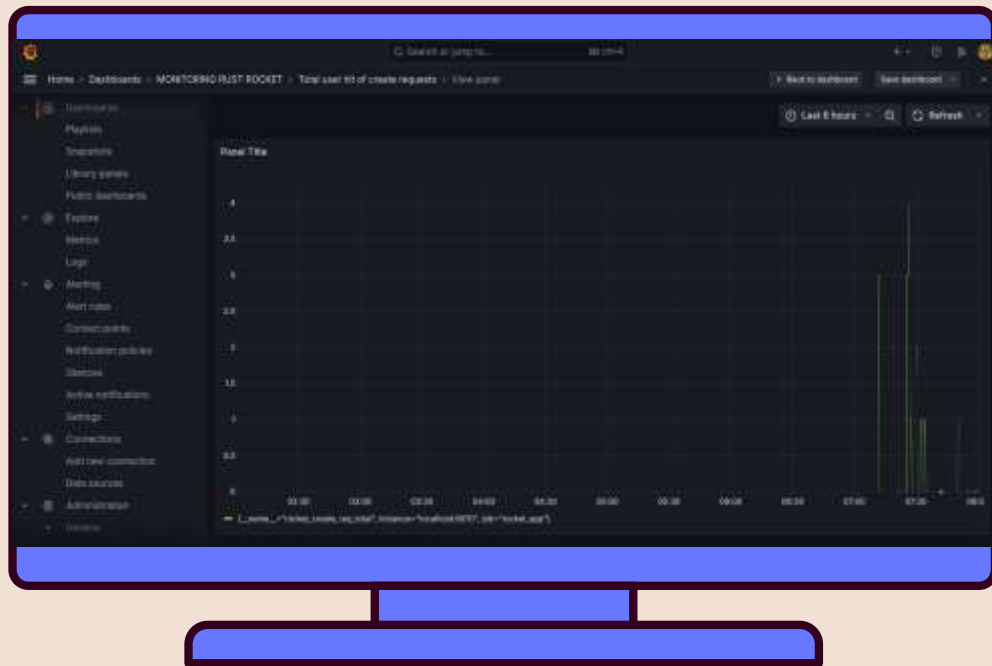
    // ...
}
```



# Create a file named metrics.rs

```
1 use prometheus::{Counter, Registry};
2
3 pub struct AppMetrics {
4     pub create_counter: Counter,
5     pub read_counter: Counter,
6     pub update_counter: Counter,
7     pub delete_counter: Counter,
8 }
9
10 impl AppMetrics {
11     pub fn new(registry: &Registry) -> Self {
12         let create_counter = Counter::new("rocket_create_req_total", "Total user hit of create requests").unwrap();
13         let read_counter = Counter::new("rocket_read_req_total", "Total user hit of create requests").unwrap();
14         let update_counter = Counter::new("rocket_update_req_total", "Total user hit of update requests").unwrap();
15         let delete_counter = Counter::new("rocket_delete_req_total", "Total user hit of delete requests").unwrap();
16
17         registry.register(Box::new(create_counter.clone())).unwrap();
18         registry.register(Box::new(read_counter.clone())).unwrap();
19         registry.register(Box::new(update_counter.clone())).unwrap();
20         registry.register(Box::new(delete_counter.clone())).unwrap();
21
22         Self {
23             create_counter,
24             read_counter,
25             update_counter,
26             delete_counter,
27         }
28     }
29 }
30
31 .....
32 .....
```

# GRAFANA MOCKUP





**SEE YOU**

.....