Assume you are expert in k6(Performance testing under javascript) and a good guide and used it in real world projects then provide a best roadmap from beginner to advance with the stages divided

Absolutely, Routh Kiran! Since you want to master **K6 for JavaScript-based Performance Testing**, here's a **well-structured roadmap** that takes you from **beginner** to **expert level**, broken into **clear stages**. This is based on real-world usage of K6 in performance testing pipelines.

🚀 Stage 1: Beginner – K6 Fundamentals

🎯 Goal: Understand what K6 is and how to write basic performance scripts.

✅ Topics:

- What is K6? Why K6 for performance testing?
- Installing K6 (via Homebrew, Chocolatey, Docker, etc.)
- CLI Basics (`k6 run`, `k6 archive`, `k6 inspect`)

First Script:

```
 import http from 'k6/http';
import { check } from 'k6';

export default function () {
  let res = http.get('https://test-api.com');
  check(res, {
    'status is 200': (r) => r.status === 200,
  });
}
```

-

- VUs (Virtual Users) vs Iterations
- Options block – setting up:
    - `vus`
    - `duration`
    - `iterations`
- Basic checks and thresholds

🎯 **Milestone: Run a basic test with VUs and thresholds.**

🧗 Stage 2: Intermediate – Load Patterns & Modularization

🎯 Goal: Handle real-world testing patterns and organize your code.

✅ **Topics:**

- Load stages (`stages`, `ramping-arrival-rate`)
- Different execution scenarios (`scenarios` block)
- HTTP methods: GET, POST, PUT, DELETE with payloads
- Modularizing test code (splitting files)
- Handling JSON APIs
- Custom headers & authentication (Bearer tokens, Basic Auth)

Setup & Teardown stages:

```
 export function setup() { ... }
export function teardown(data) { ... }
```

-

- `k6/html` for HTML scraping (optional)

🎯 Milestone: Modularize a test plan and simulate staged load.

🔧 Stage 3: Scripting Advanced Logic

🎯 Goal: Build dynamic, realistic, and stateful test flows.

✅ Topics:

- Dynamic test data using `__ITER` or UUIDs

Data-driven testing (CSV, JSON):

```
import { SharedArray } from 'k6/data';
```

-
- Randomized data and test users
- Cookie and Session Management
- Custom metrics (Counter, Gauge, Trend, Rate)

Grouping and nested groups:

```
group("Login flow", function() { ... });
```

-

🎯 Milestone: Simulate full user journeys with dynamic data.

📊 Stage 4: Analysis, Reporting & Debugging

🎯 Goal: Analyze performance effectively with proper tools.

✅ Topics:

- Understanding built-in metrics:

    - `http_req_duration`
    - `http_req_failed`
    - `iterations`, `vus`
- Thresholds for pass/fail logic
- Using `k6 stats`, `k6 run --summary-export`
- Output types:
    - JSON
    - InfluxDB + Grafana Dashboard
    - Prometheus remote write
- k6 Cloud (optional)
- Debug mode (`--http-debug`)

🎯 **Milestone: Create a dashboard with InfluxDB + Grafana for test reports.**

🌐 Stage 5: Real World Integration

🎯 Goal: Use K6 in CI/CD and build test pipelines.

✅ **Topics:**

- Running K6 in:
    - Jenkins
    - GitHub Actions
    - GitLab CI
    - Dockerized pipelines

- Test environments and mocking
- Post-test alerting via email or Slack (using thresholds + CI)
- Using K6 extensions (xk6)
    - gRPC, Kafka, Redis, etc.

🎯 **Milestone: Integrate K6 tests in your CI/CD with test pass/fail logic.**

🧠 Stage 6: Expert Level Mastery

🎯 Goal: Contribute, extend and fully own performance testing strategy.

✅ **Topics:**

- Writing custom extensions using Go (xk6)
- Simulating real traffic patterns (arrival rate, duration-based)
- Advanced metrics aggregation and correlation
- Load distribution testing (multi-region, cloud)
- Performance tuning recommendations (server-side)
- Contributing to K6 open-source or creating plugins
- Best practices in test script versioning, logging, and documentation

🎯 **Milestone: Become the go-to performance engineer in your team.**

🛠️ Tools You'll Use Along the Way

- K6 CLI
- Grafana + InfluxDB/Prometheus

- Docker & GitHub Actions
- Visual Studio Code + REST clients
- Postman (for debugging APIs before scripting)

📚 Suggested Practice Projects

1. **Test a real API** – mock API or public APIs like `reqres.in`
2. **Simulate e-commerce flow** – login, search, add-to-cart, checkout
3. **Create a CI-integrated load test** – GitHub Actions + Grafana
4. **Build reusable test framework** – modular & configurable via JSON/YAML

Would you like a **Notion board or PDF roadmap version** of this as well?

Let me know if you want **code examples** or **CI templates** to speed up your learning!