

Supabase Private Command Center

Self-hosted application backend with private administration and integrated operations

Customer Launch & Operations Guide | Code Creator

WHY CODE CREATOR

Run your application backend in your own AWS account - without assembling the stack and operating workflow from scratch. Code Creator automates first-launch initialization, creates instance-specific configuration, checks application readiness, and adds a live operations dashboard with integrated backup, recovery, and diagnostics.

Supabase Private Command Center by Code Creator brings together PostgreSQL, authentication, APIs, file storage, realtime capabilities, and Edge Functions in a self-hosted AWS appliance. Your applications connect to the public API endpoint. Administrative access to the Code Creator Command Center and Supabase Studio stays behind the local interface and is normally reached through an authenticated SSH tunnel using your authorized EC2 SSH access.

1. Why customers choose this appliance

- Less setup work: launch an integrated backend with automated first-launch initialization and instance-specific configuration.
- Clear operational visibility: check service health, application readiness, CPU, memory, disk use, and next steps from one dashboard.
- Private administration: keep the Command Center and Supabase Studio off the public application endpoint.
- Practical recovery tools: create full-backend backups and restore validated backend state through integrated workflows.
- Infrastructure control: choose and manage the AWS resources that host your application backend and data.

YOUR BACKEND. YOUR AWS ACCOUNT.

Designed for developers and small engineering teams that want control over their backend infrastructure and data, with practical tools for running it inside their AWS account.

OPERATING EXPECTATIONS

This is a customer-operated, single-instance appliance. Customers remain responsible for capacity planning, application security policies, trusted production TLS, maintenance, and off-instance backups. High availability and automatic scaling are not included. Applicable AWS infrastructure charges are separate from the software charge.

2. Launch requirements

Validated AWS configuration	Fresh-launch validation passed on t3.xlarge. Use the Marketplace-supported instance guidance for your workload.
Validated root volume	50 GiB was used for the release validation instance. This proves the release, not production capacity.
Recommended evaluation starting point	100 GiB gp3 is a safer starting point for evaluation so

	database growth, uploaded objects, image layers, backups, and restore working space do not immediately compete for the same disk.
Production capacity	Size CPU, memory, EBS capacity, throughput, and IOPS for database growth, object storage, concurrency, backup size, and recovery windows. A restore can temporarily require several times the live dataset size.
Inbound security group	TCP 22 (SSH), TCP 80 (HTTP), TCP 443 (HTTPS). Do not expose 8443, 5432, 6543, 8000, or 8001.
EC2 key pair	Select a key pair you control. Administrative access normally uses authenticated SSH access to the Ubuntu host.
SSH user	ubuntu

3. First launch

1. Launch the AWS Marketplace AMI and wait for the EC2 instance to become reachable. EC2 status checks completing does not mean the application initialization has finished.
2. Connect to the instance as ubuntu using the EC2 key pair selected at launch.

```
ssh -i /path/to/your-key.pem ubuntu@PUBLIC_IP
```

3. Check for the generated first-launch information:

```
cat ~/FIRST_LOGIN.txt
```

FIRST_LOGIN.txt is created only after the appliance first-boot process finishes. If the file does not exist yet, check firstboot instead of assuming the launch failed:

```
sudo systemctl status codecreator-firstboot --no-pager
```

```
sudo journalctl -u codecreator-firstboot -n 100 --no-pager
```

4. Confirm backend health after firstboot completes:

```
sudo codecreator-supabase health
```

A successful launch reports overall PASS and healthy status for all 11 appliance services. Some services can briefly report starting or unhealthy while dependencies converge; firstboot waits for the stack to become healthy before completing.

4. Public application endpoint and API keys

PUBLIC APPLICATION ENDPOINT

https://PUBLIC_IP

Use this origin for application-facing Supabase APIs. Your application does not use the administrative SSH tunnel.

Typical application-facing paths are available below the public HTTPS origin, including Auth, REST, Storage, Realtime, and Edge Functions:

```
https://PUBLIC\_IP/auth/v1/
```

```
https://PUBLIC\_IP/rest/v1/
```

```
https://PUBLIC\_IP/storage/v1/
```

https://PUBLIC_IP/functions/v1/

Application authorization

Network reachability does not make application data anonymous or unrestricted. Obtain the project publishable key from Supabase Studio (Project Settings / API Keys or the project connection panel) and use it for client-side application requests as appropriate.

- Use the publishable key in browser/mobile clients. Do not place the secret key, service-role key, or other privileged credentials in client-side code.
- API requests commonly send the project key in the apikey header. Authenticated user requests also use the user session access token as an Authorization: Bearer token where required.
- Configure PostgreSQL Row Level Security, Storage policies, Auth settings, and application authorization rules for your workload. Public HTTPS access is only the network path; access policies still control data operations.

Production TLS for a DNS name

The first-launch public endpoint uses an instance-generated certificate. Browser and mobile clients may reject a self-signed certificate. For production, use a stable DNS name and a trusted certificate.

1. Point the production DNS name to the stable AWS endpoint you intend to keep, such as an Elastic IP or another approved stable network design.
2. Obtain a trusted certificate and matching private key in PEM format for that DNS name. Keep the private key protected on the Ubuntu host.
3. Apply the certificate using the appliance-supported endpoint tool. Example:

```
sudo python3 /opt/codecreator-supabase/scripts/set_endpoint.py \
  --origin https://backend.example.com \
  --certificate /home/ubuntu/server.crt \
  --private-key /home/ubuntu/server.key \
  --confirm
```

The appliance verifies the certificate hostname, expiration, and private-key match; updates the public Supabase URLs; installs the TLS material; validates nginx configuration; and restarts the affected services.

4. Verify the deployed certificate and appliance health:

```
openssl s_client -connect backend.example.com:443 -servername backend.example.com </dev/null
2>/dev/null \
  | openssl x509 -noout -subject -issuer -dates -fingerprint -sha256
sudo codecreator-supabase health
```

CERTIFICATE RENEWAL

Certificate renewal is the customer's responsibility. Before the trusted certificate expires, obtain the renewed certificate/key and run the same set_endpoint.py command again with the new files. Do not manually edit the generated nginx site configuration; appliance startup can regenerate that configuration from the installed source.

5. Private administrative access

The Code Creator Command Center and Supabase Studio are intentionally not exposed on the public IP. They are bound to the local administrative interface and are normally reached through an authenticated SSH tunnel using your authorized SSH access to the EC2 host.

Command Center: <https://127.0.0.1:8443/cc/>

Supabase Studio: <https://127.0.0.1:8443/>

Keep the SSH tunnel open while using either administrative interface. Public requests to these administrative paths are expected to return 404 Not Found. Users with other authorized privileged host access can also reach localhost services; protect all authorized host access accordingly.

macOS

macOS includes OpenSSH. Open Terminal. If required, restrict private-key permissions:

```
chmod 400 ~/Downloads/your-key.pem
```

Start the tunnel:

```
ssh -N -i ~/Downloads/your-key.pem \
-o ExitOnForwardFailure=yes \
-L 127.0.0.1:8443:127.0.0.1:8443 \
ubuntu@PUBLIC_IP
```

Linux

Most Linux distributions include OpenSSH. If required, restrict private-key permissions:

```
chmod 400 /path/to/your-key.pem
```

Start the tunnel:

```
ssh -N -i /path/to/your-key.pem \
-o ExitOnForwardFailure=yes \
-L 127.0.0.1:8443:127.0.0.1:8443 \
ubuntu@PUBLIC_IP
```

Windows with OpenSSH

Windows systems with OpenSSH installed can use PowerShell or Windows Terminal:

```
ssh -N -i "C:\path\to\your-key.pem" `
-o ExitOnForwardFailure=yes `
-L 127.0.0.1:8443:127.0.0.1:8443 `
ubuntu@PUBLIC_IP
```

Windows with PuTTY

1. Open PuTTY and enter the EC2 public IP under Session > Host Name.
2. Under Connection > SSH > Auth, select the **.ppk key** that corresponds to the EC2 key pair used at launch.
3. Go to **Connection > SSH > Tunnels**
4. Set Source port to **8443** and Destination to **127.0.0.1:8443**
5. Select **Local**. Do not allow other hosts to connect to the local port. Click **Add**
6. Confirm the rule appears as L8443 127.0.0.1:8443, return to Session, and open the SSH connection as **ubuntu**

6. Administrative TLS verification

Because the administrative site is opened in the browser as 127.0.0.1 through the SSH tunnel, a browser certificate warning can occur. Verify the instance-generated administrative certificate fingerprint over your authenticated SSH session:

```
sudo openssl x509 \
-in /etc/codecreator-supabase/tls/admin.crt \
-noout -fingerprint -sha256
```

Compare the displayed SHA-256 fingerprint with the certificate shown by your browser before proceeding when certificate verification is required by your environment.

7. Code Creator Command Center

Open: <https://127.0.0.1:8443/cc/>

The Command Center is the operational front door to the appliance. It reduces the Linux and Docker work normally required to understand whether a self-hosted backend is ready and healthy.

- View service health, CPU, memory, disk capacity, uptime, application readiness, backup state, and operational alerts.
- Use Run readiness checks to verify application-level connectivity in addition to container health.
- Use Guided setup to configure the application redirect URL and SMTP settings.
- Use Backup & recovery and Diagnostics for integrated operational workflows.
- Enable email signups only after SMTP is configured and verified.

8. Supabase Studio

Open: <https://127.0.0.1:8443/>

Supabase Studio provides the self-hosted project development and administration workspace. Release validation confirmed the project overview and core routes for SQL, database, authentication, storage, Edge Functions, and Realtime through the private administrative tunnel.

SELF-HOSTED FEATURE EXPECTATIONS

A visible Studio menu does not prove that every Supabase managed-platform feature is available or validated in this appliance. Self-hosted Supabase differs from the managed Supabase platform. Managed features such as branching, managed backups/PITR, the Management API, and some platform analytics/observability capabilities are not included by this appliance. See <https://supabase.com/docs/guides/self-hosting> for current upstream self-hosting differences.

Supabase Studio is intentionally unavailable through the public EC2 IP.

9. Operations CLI

Product information	<code>sudo codecreator-supabase info</code>
Status	<code>sudo codecreator-supabase status</code>
Full health	<code>sudo codecreator-supabase health</code>
Create local backup	<code>sudo codecreator-supabase backup --confirm</code>
Restart backend	<code>sudo codecreator-supabase restart --confirm</code>
Diagnostics	<code>sudo codecreator-supabase diagnostics</code>
Security check	<code>sudo codecreator-supabase security</code>
CLI reference	<code>codecreator-supabase --help</code>

ABOUT THE SECURITY CHECK

The security command checks local secret-file permissions and basic secret initialization. It is not a CVE scanner, penetration test, compliance assessment, or certification of the operating system and containers.

10. Backup and recovery

The appliance includes an integrated full-backend backup workflow. The validated recovery path restores database records, uploaded storage objects, configuration/environment values, and application health.

BACKUPS REQUIRE DOWNTIME

Creating a full appliance backup stops backend services to create a consistent recovery point. Schedule backups during an acceptable maintenance window. This appliance does not provide continuous backup or managed Point-in-Time Recovery (PITR).

BACKUP LOCATION MATTERS

A backup stored only on the same EC2 volume does not protect against complete instance or volume loss. For production workloads, maintain an additional recovery copy outside the instance using your approved AWS backup strategy.

Create a local backup and explicitly confirm the downtime:

```
sudo codecreator-supabase backup --confirm
```

- Create and verify backups before significant application or infrastructure changes.
- Restore is destructive. Select the intended recovery archive carefully and use the appliance confirmation controls.
- After recovery, validate database records, uploaded objects, authentication, configuration, and application health.

The appliance also includes optional S3 configuration, upload, and scheduling operations. Review the installed CLI interface before configuring these features:

```
codecreator-supabase --help
```

Where AWS access is required, prefer an appropriately scoped EC2 IAM role instead of storing long-lived AWS access keys on the server. Scheduled full backups also require downtime.

11. Database and internal network access

PostgreSQL, Supavisor, internal gateways, and the administrative listener are intentionally bound to localhost. Do not expose these services directly to the Internet.

When direct database access is required, use SSH port forwarding rather than opening PostgreSQL publicly.

12. Security model

- Application-facing Supabase APIs are available through the public application endpoint, subject to API keys, user authentication, Row Level Security, Storage policies, and other application controls you configure.
- Administrative access is restricted to the local interface and is normally reached through an authenticated SSH tunnel using your EC2 launch key. Other authorized privileged host access can also reach localhost services.
- Administrative web paths return 404 when requested through the public IP.
- MCP is not publicly exposed. The public MCP path is unavailable and administrative access is denied by policy.
- Do not place secret, service-role, or other privileged keys in browser-side applications.
- Protect all SSH credentials and privileged host access. They are part of the administrative trust boundary.

13. Public IP changes

A normal stop/start of the same EC2 instance may result in a different public IPv4 address. The appliance tracks EC2 instance identity separately from the public address and is designed to preserve the same-instance trust boundary while updating generated IP-based network configuration.

For a stable public endpoint, use an appropriate AWS networking and DNS design, such as an Elastic IP or another stable address, together with a trusted certificate. When using customer-supplied TLS for a DNS name, renew and reapply the certificate before expiration.

14. Troubleshooting

FIRST_LOGIN.txt is missing	EC2 status checks can finish before application initialization. Run <code>sudo systemctl status codecreator-firstboot --no-pager</code> and <code>sudo journalctl -u codecreator-firstboot -n 100 --no-pager</code> .
Command Center or Studio does not open	Confirm the SSH session is still open and local port 8443 is forwarded to 127.0.0.1:8443.
Public /cc/ or Studio returns 404	Expected. Administrative paths are private by design.
Browser certificate warning	Expected for the instance-generated administrative certificate. Verify its SHA-256 fingerprint over SSH.
Public application TLS warning	Use the supported <code>set_endpoint.py</code> procedure to install a trusted certificate for your production DNS name.
Service not ready	Run <code>sudo codecreator-supabase health</code> and inspect the affected service. First launch can require time for dependencies to converge.
Application signup warning	Complete the redirect URL and SMTP configuration before enabling application signups.

15. Important operational notes

- Do not expose TCP 8443, 5432, 6543, 8000, or 8001 directly to the Internet.
- Do not manually edit generated nginx configuration to install production TLS; use the supported endpoint/certificate tool so the appliance configuration stays consistent.
- Do not manually modify appliance identity or first-boot files.
- Do not use `rearm-for-ami` or `bind-instance` during normal customer operation unless specifically instructed by Code Creator support.
- Keep AWS security groups restrictive and permit only traffic required by your application.
- Maintain tested off-instance backups and plan maintenance windows for full backups and restores.
- Plan capacity for database growth, uploaded objects, container/image overhead, backups, and temporary restore space.

16. Quick reference

Public application endpoint	<code>https://PUBLIC_IP</code>
------------------------------------	--------------------------------

Command Center	<code>https://127.0.0.1:8443/cc/</code> (SSH tunnel required)
Supabase Studio	<code>https://127.0.0.1:8443/</code> (SSH tunnel required)
SSH user	<code>ubuntu</code>
First-launch information	<code>cat ~/FIRST_LOGIN.txt</code>
Firstboot status	<code>sudo systemctl status codecreator-firstboot --no-pager</code>
Firstboot log	<code>sudo journalctl -u codecreator-firstboot -n 100 --no-pager</code>
Health	<code>sudo codecreator-supabase health</code>
Status	<code>sudo codecreator-supabase status</code>
Product information	<code>sudo codecreator-supabase info</code>
Backup	<code>sudo codecreator-supabase backup --confirm</code>
Security check scope	Local permissions/basic secret initialization only; not a vulnerability scan

CODE CREATOR

Your backend. Your AWS account.

Supabase Private Command Center by Code Creator is a self-hosted application backend with private administration and integrated operations - automated first launch, live readiness and health visibility, backup/recovery workflows, and a Code Creator Command Center for operating the appliance inside your AWS account.