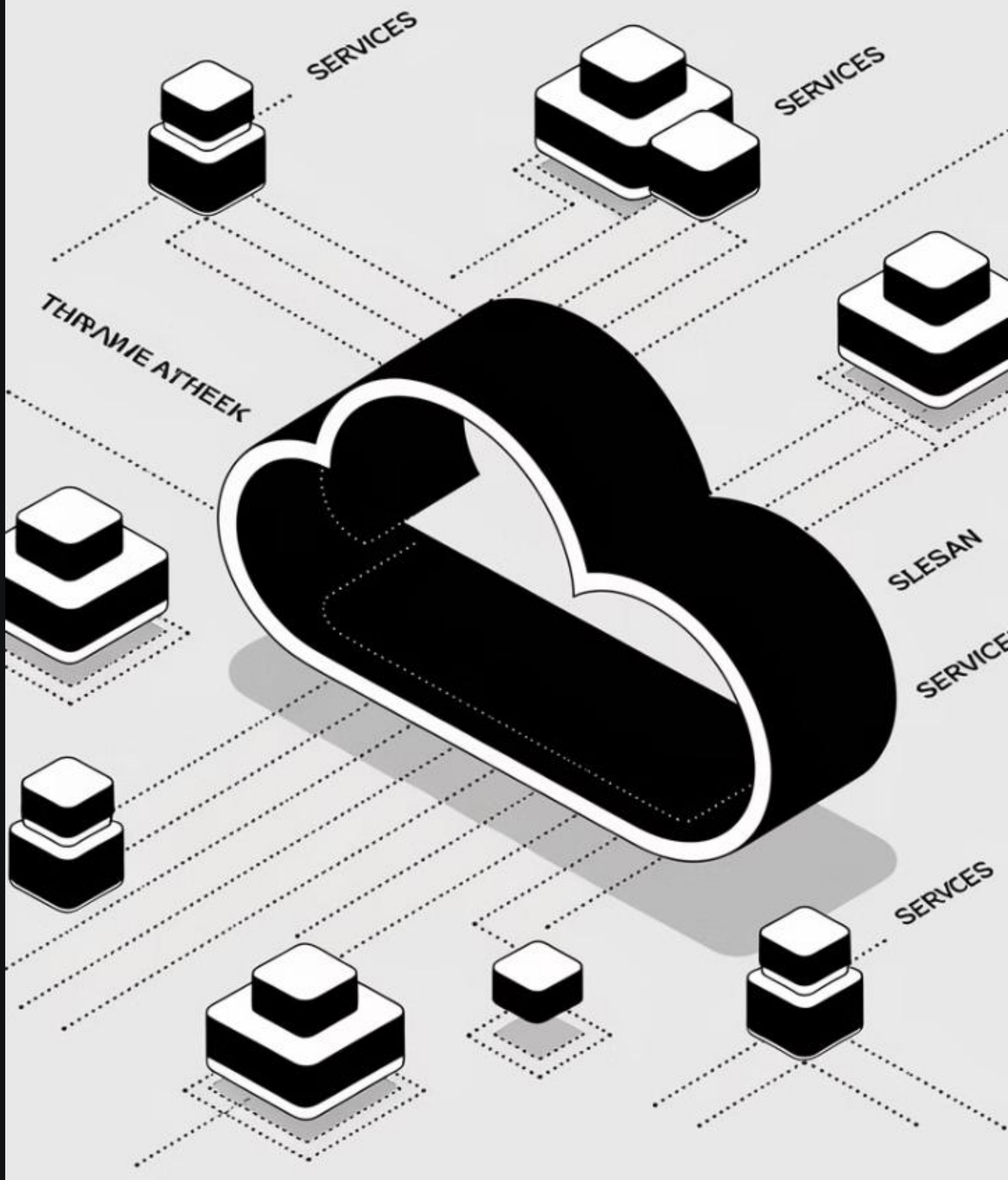
Cloud-Native Architecture in SAP BTP

por Mayko Silva



What is Cloud-Native Architecture?

Cloud-native architecture represents an approach to designing, building, and running applications that fully exploit the advantages of cloud computing. Rather than simply migrating existing applications to cloud infrastructure, cloud-native architecture reimagines applications to thrive in dynamic, distributed environments.

In the context of SAP BTP, cloud-native architecture enables organizations to:

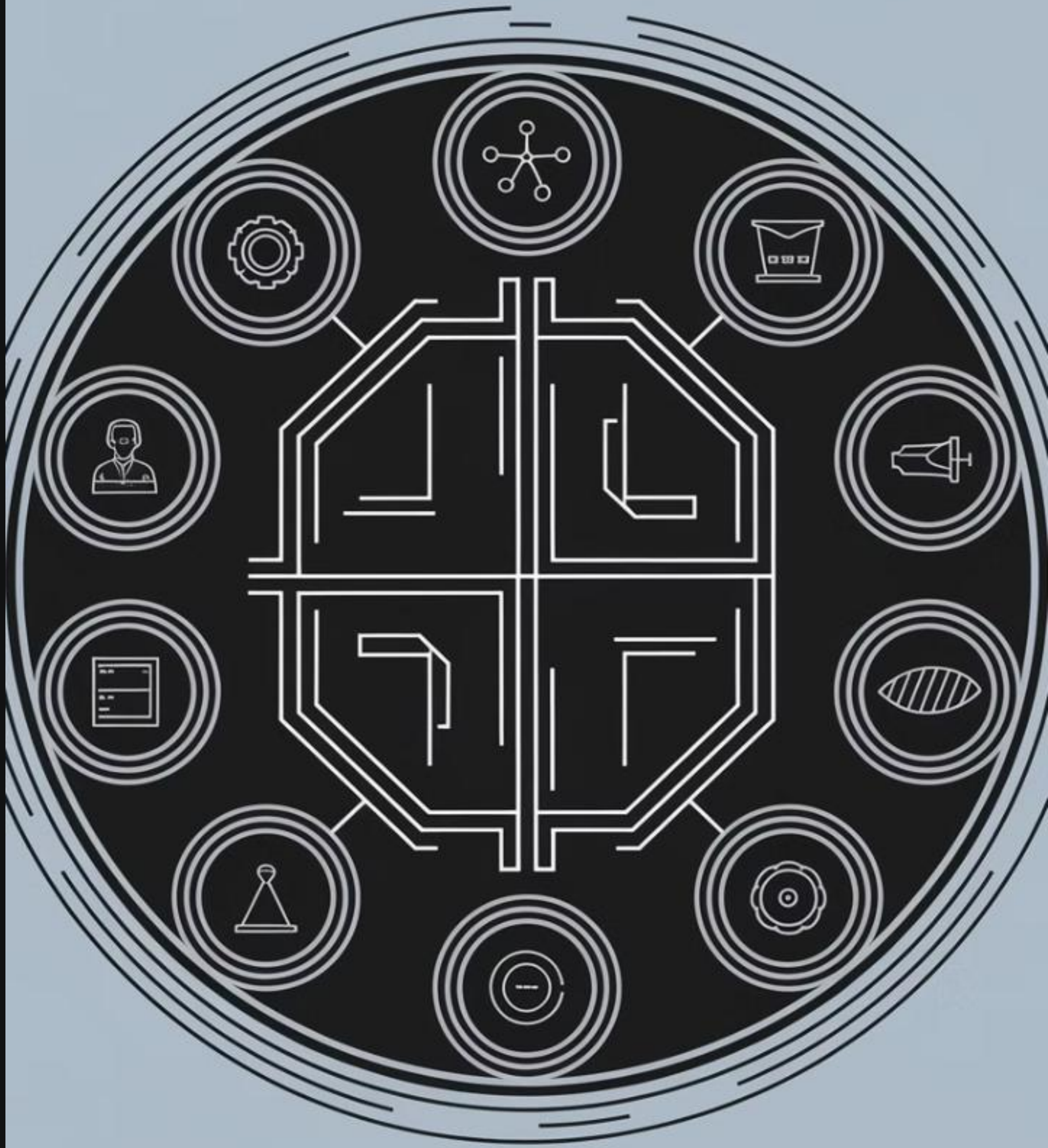
- Build highly scalable and resilient applications
- Improve development velocity and operational efficiency
- Adapt quickly to changing business requirements
- Utilize cloud resources efficiently and cost-effectively

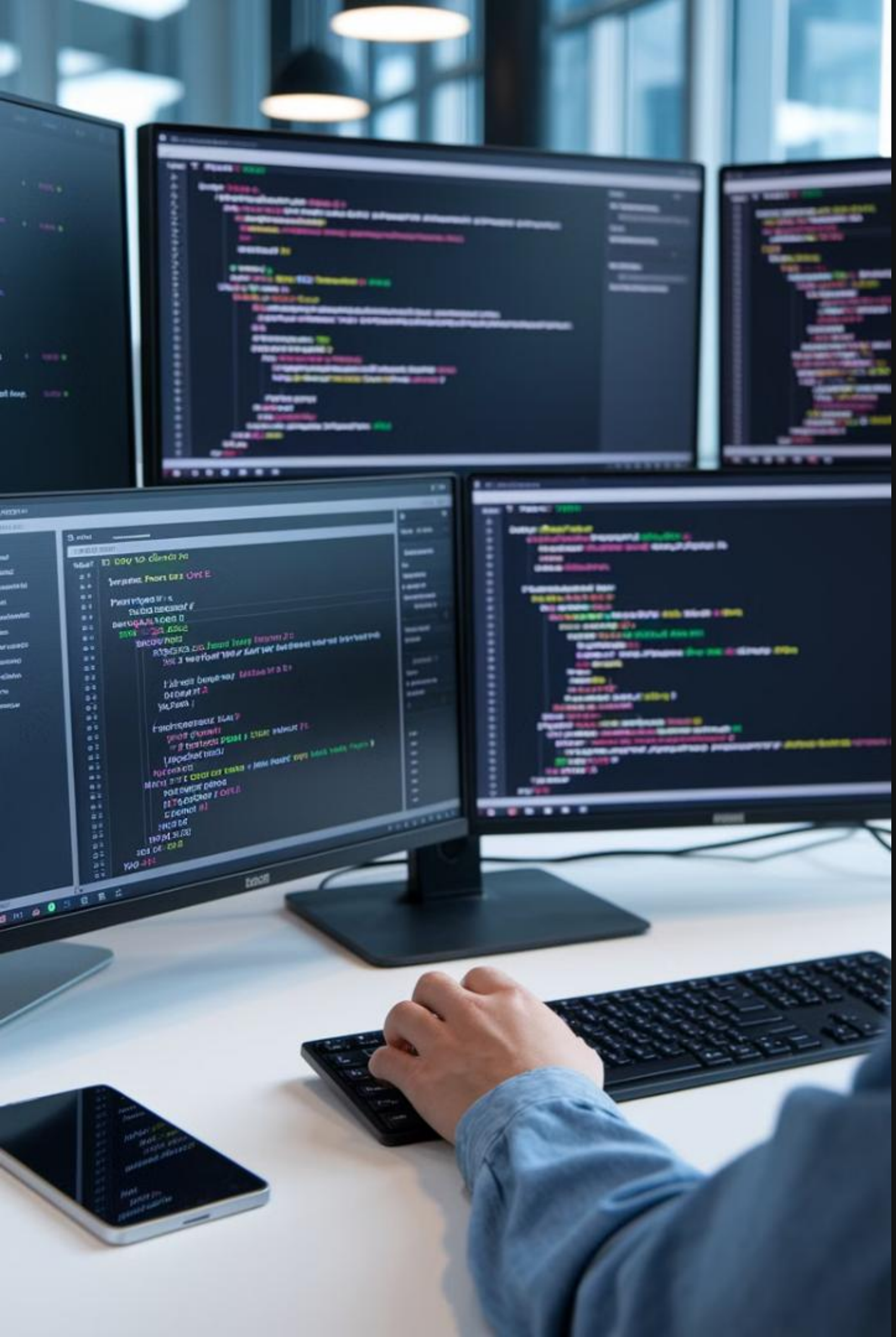
Cloud-native applications typically embrace microservices, containers, orchestration platforms, and automated DevOps practices. These applications are designed to handle the ephemeral nature of cloud resources and expect failures as a normal part of operation.

Twelve-Factor Principles

1 The Twelve-Factor App methodology provides a set of principles for building cloud-native applications that are optimized for modern cloud platforms like SAP BTP. These principles were originally formulated by developers at Heroku but have become widely adopted across the industry.

2 Let's examine these principles and how they apply to SAP BTP development:

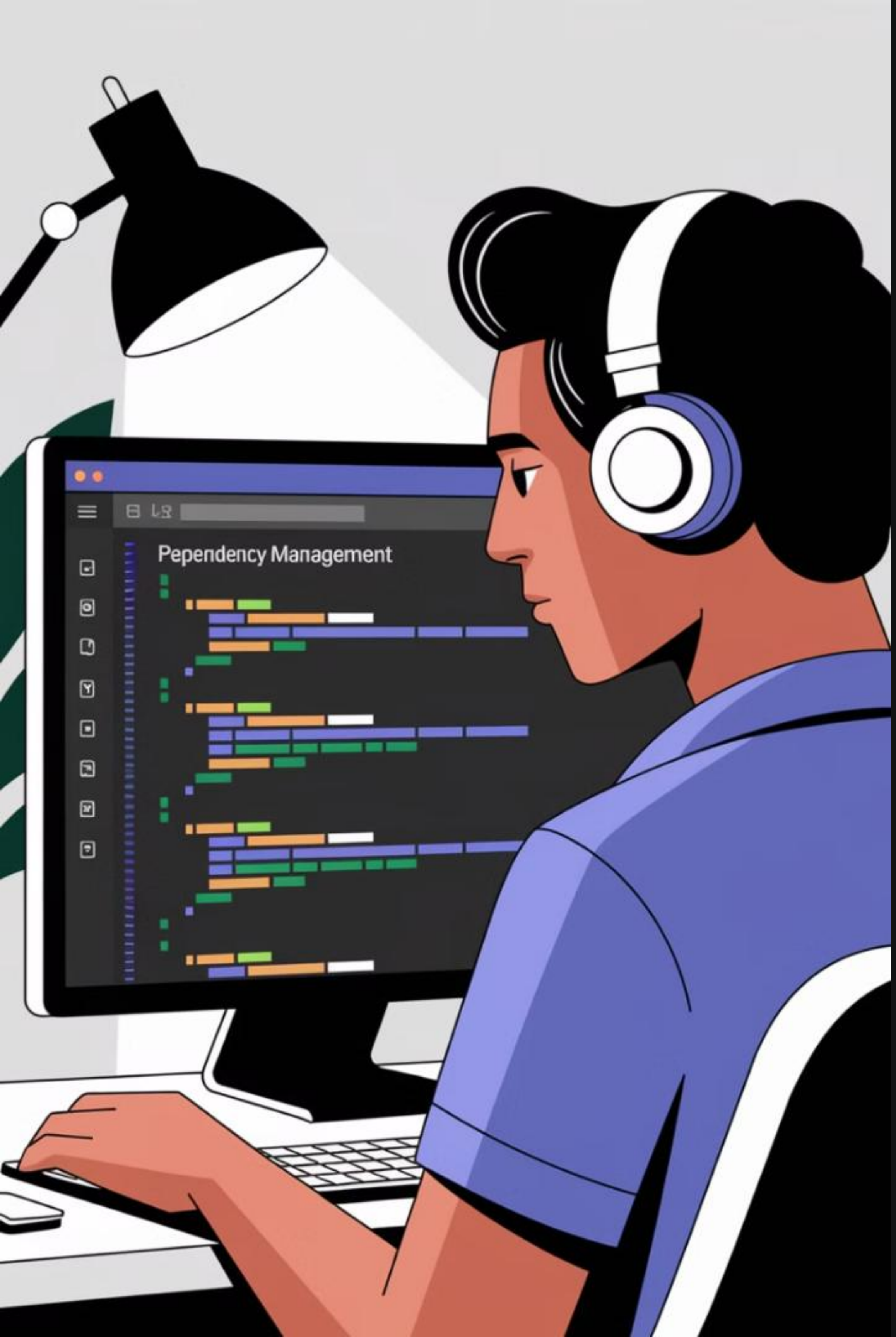




1. Codebase

One codebase tracked in version control, many deployments.

In SAP BTP, this means maintaining a single repository for each microservice or application component, with clear branching strategies for different environments (development, testing, production).



2. Dependencies

Explicitly declare and isolate dependencies.

When developing for SAP BTP, all dependencies should be explicitly declared in files like package.json (for Node.js), pom.xml (for Java), or requirements.txt (for Python). The SAP Business Application Studio provides excellent support for dependency management.

3. Configuration

Store configuration in the environment.

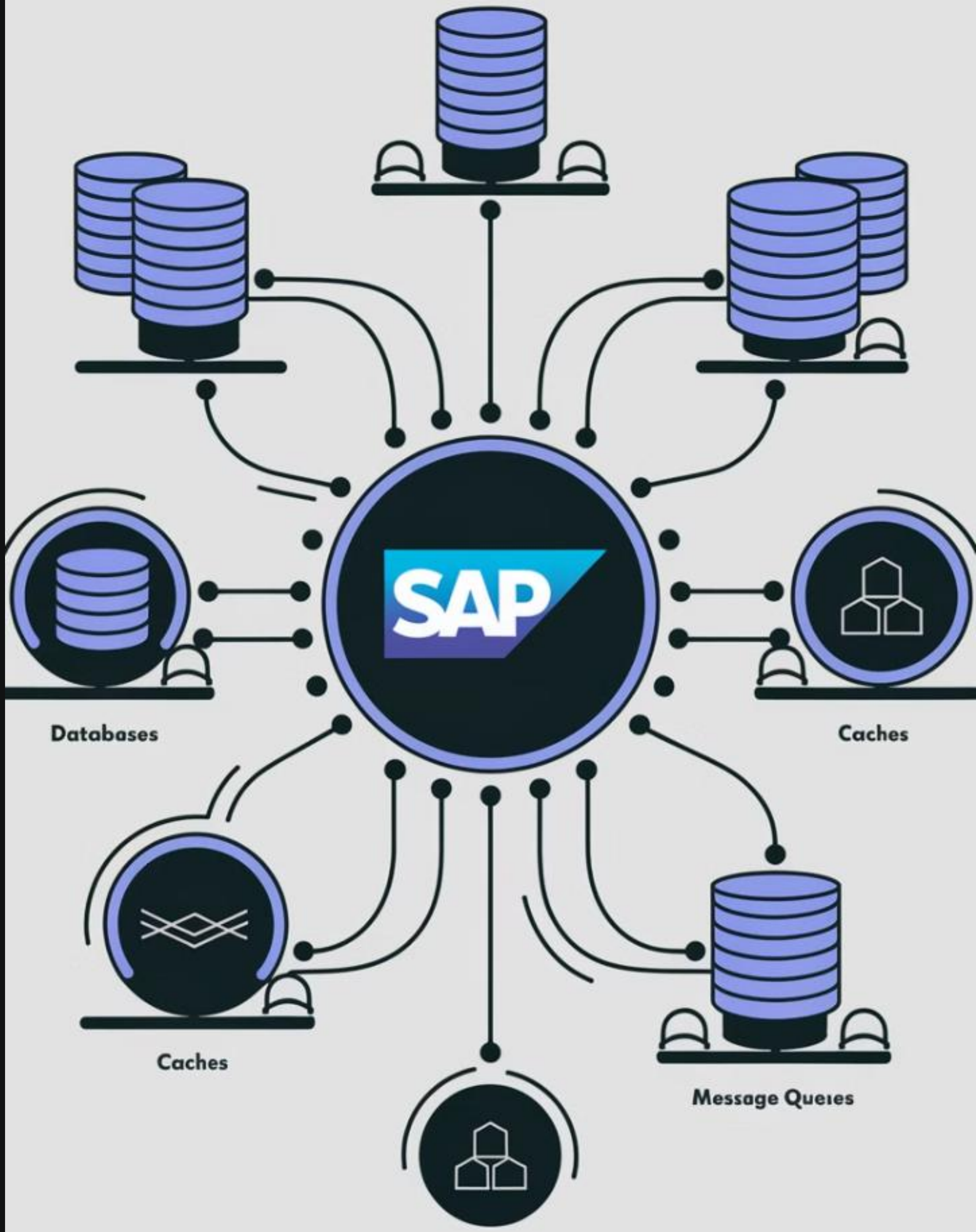
SAP BTP provides several mechanisms for environment-based configuration:

- Environment variables in Cloud Foundry
- Kubernetes ConfigMaps and Secrets in the Kyma runtime
- The Destination service for connection information
- SAP BTP Credential Store for sensitive information

4. Backing Services

Treat backing services as attached resources.

In SAP BTP, services like SAP HANA, Redis, PostgreSQL, or messaging services are provisioned and bound to applications without code changes. The application simply adapts to whatever resources are attached through service bindings.



5. Build, Release, Run

1

Build

Automated build pipelines that create immutable artifacts

2

Release

Deployment mechanisms that combine these artifacts with configuration

3

Run

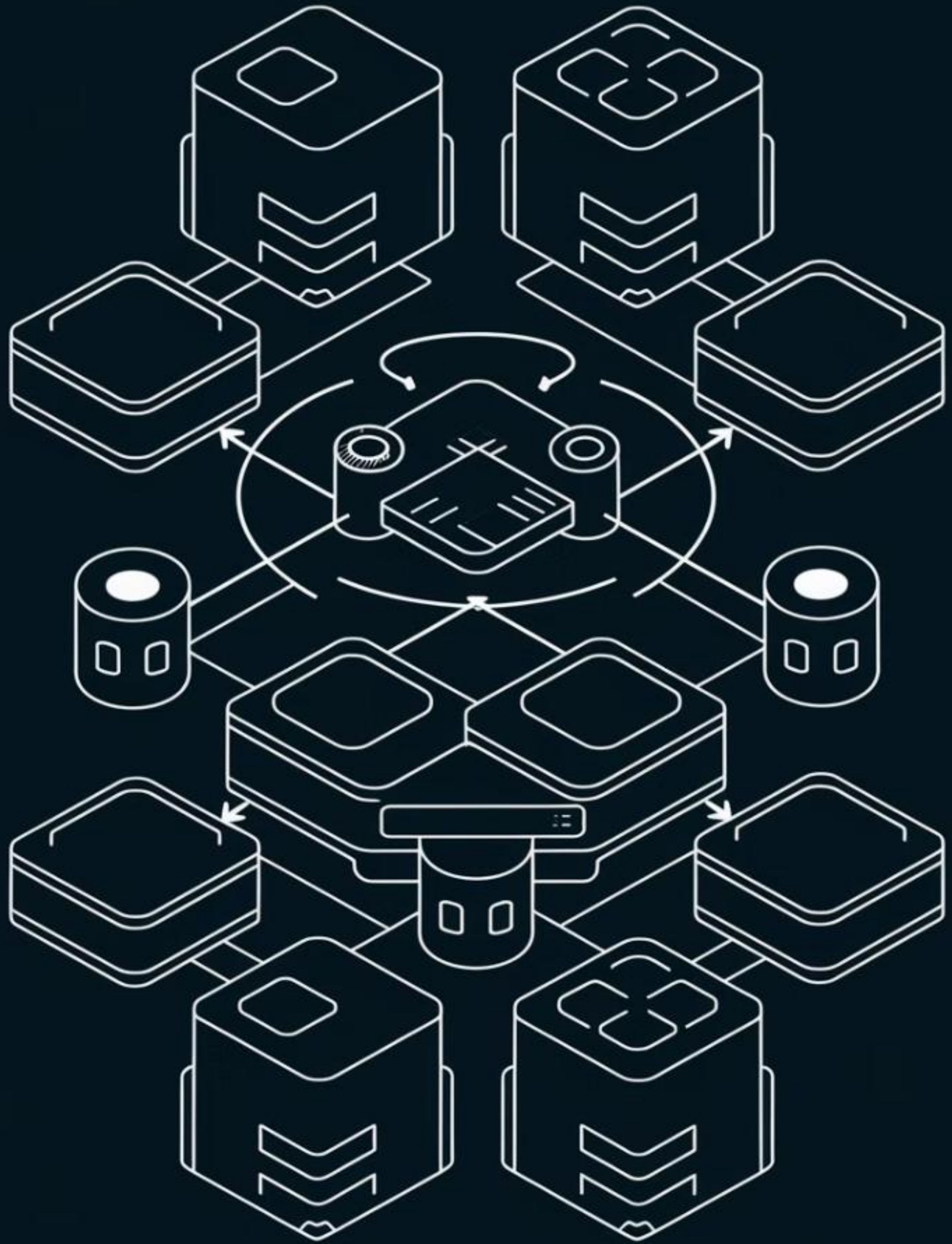
Runtime environments that execute the released version

SAP BTP's CI/CD services support this principle through these stages.

6. Processes

Execute applications as one or more stateless processes.

Applications in SAP BTP Cloud Foundry or Kyma runtimes should be designed as stateless processes. Any state should be stored in backing services like databases or cache systems.



7. Port Binding

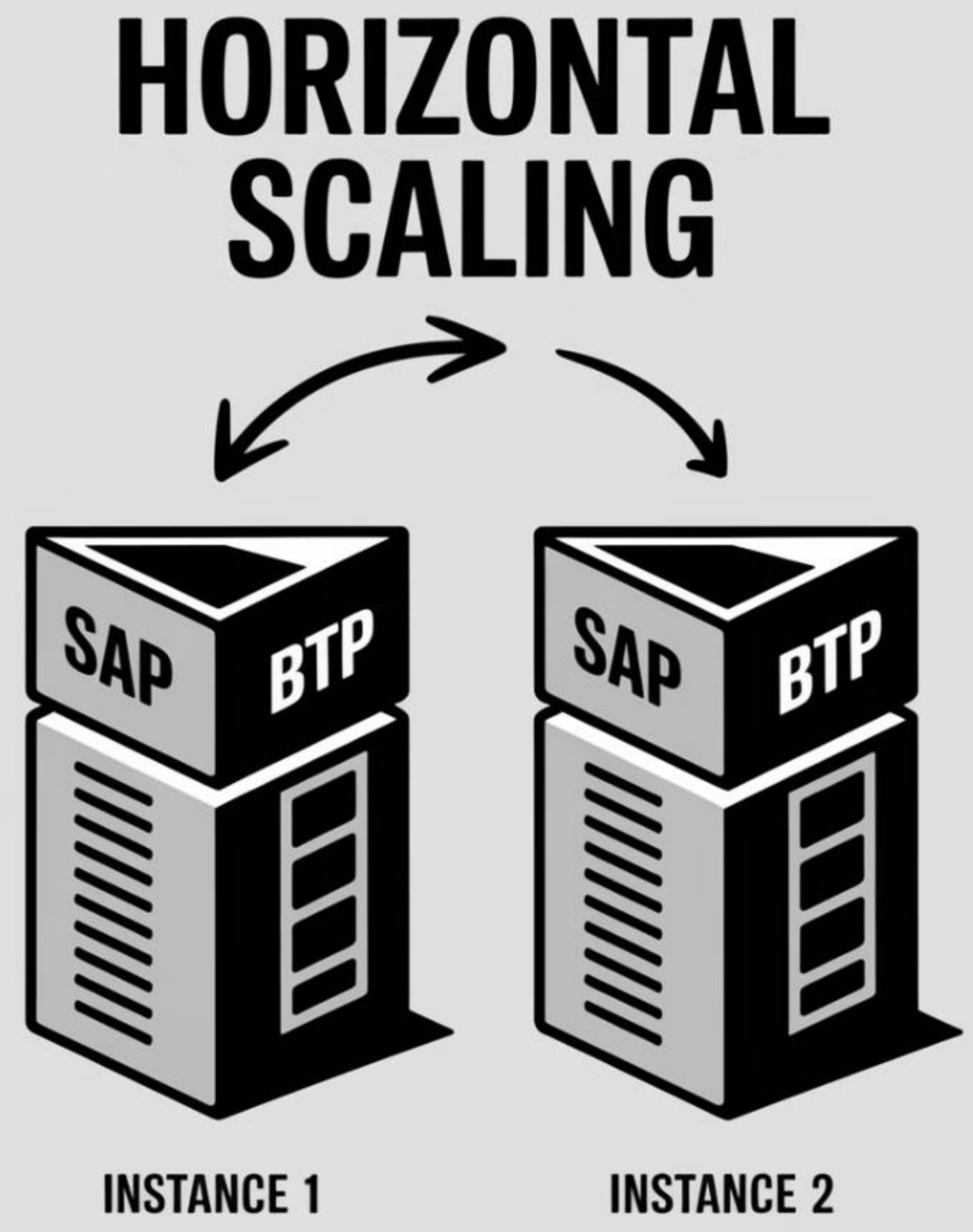
Export services via port binding.

In SAP BTP, applications self-contain their web servers and bind to ports assigned by the platform. This enables the platform to route traffic appropriately.

8. Concurrency

Scale out via the process model.

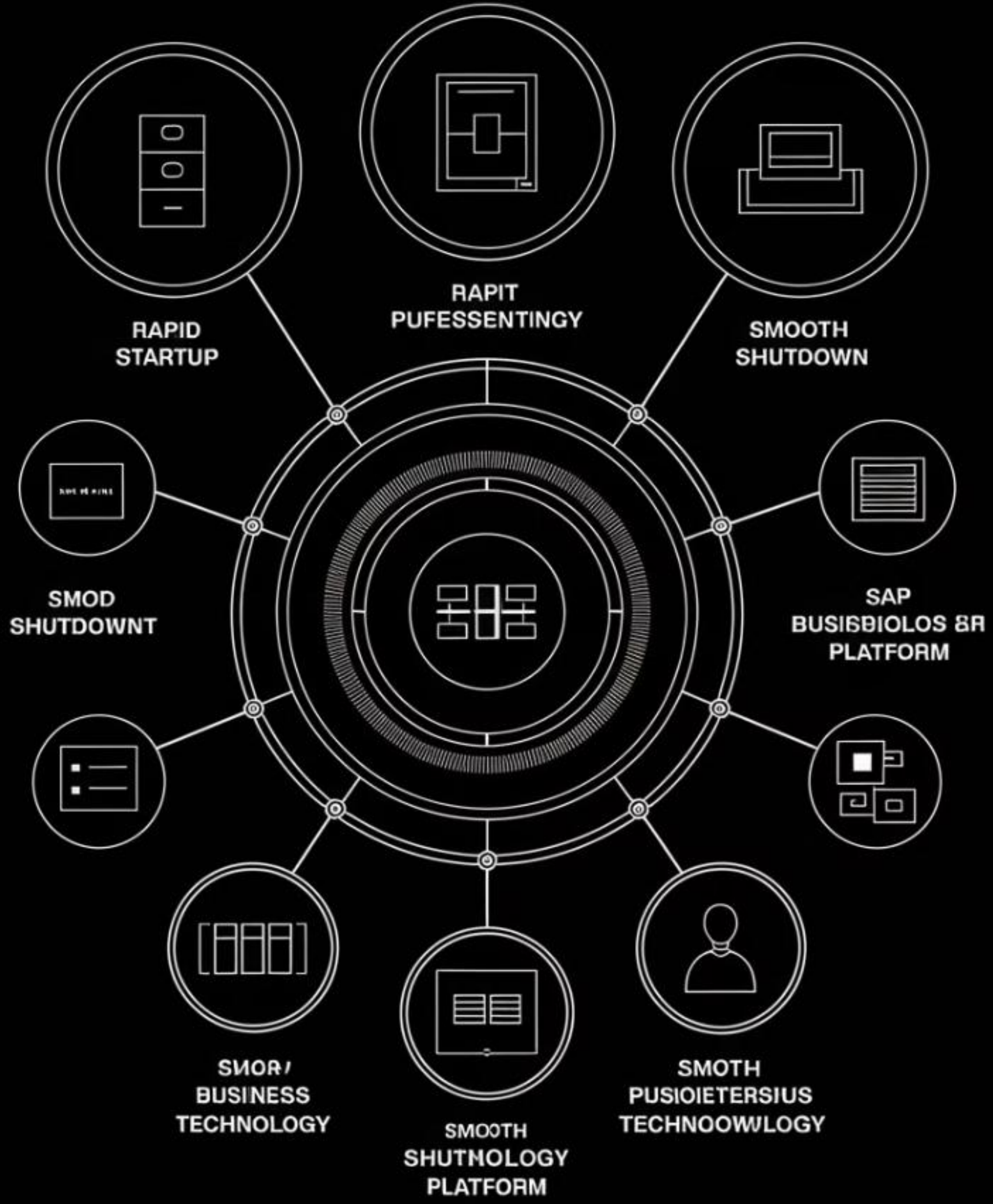
SAP BTP allows horizontal scaling by running multiple instances of applications. Applications should be designed to scale horizontally rather than vertically.



9. Disposability

Maximize robustness with fast startup and graceful shutdown.

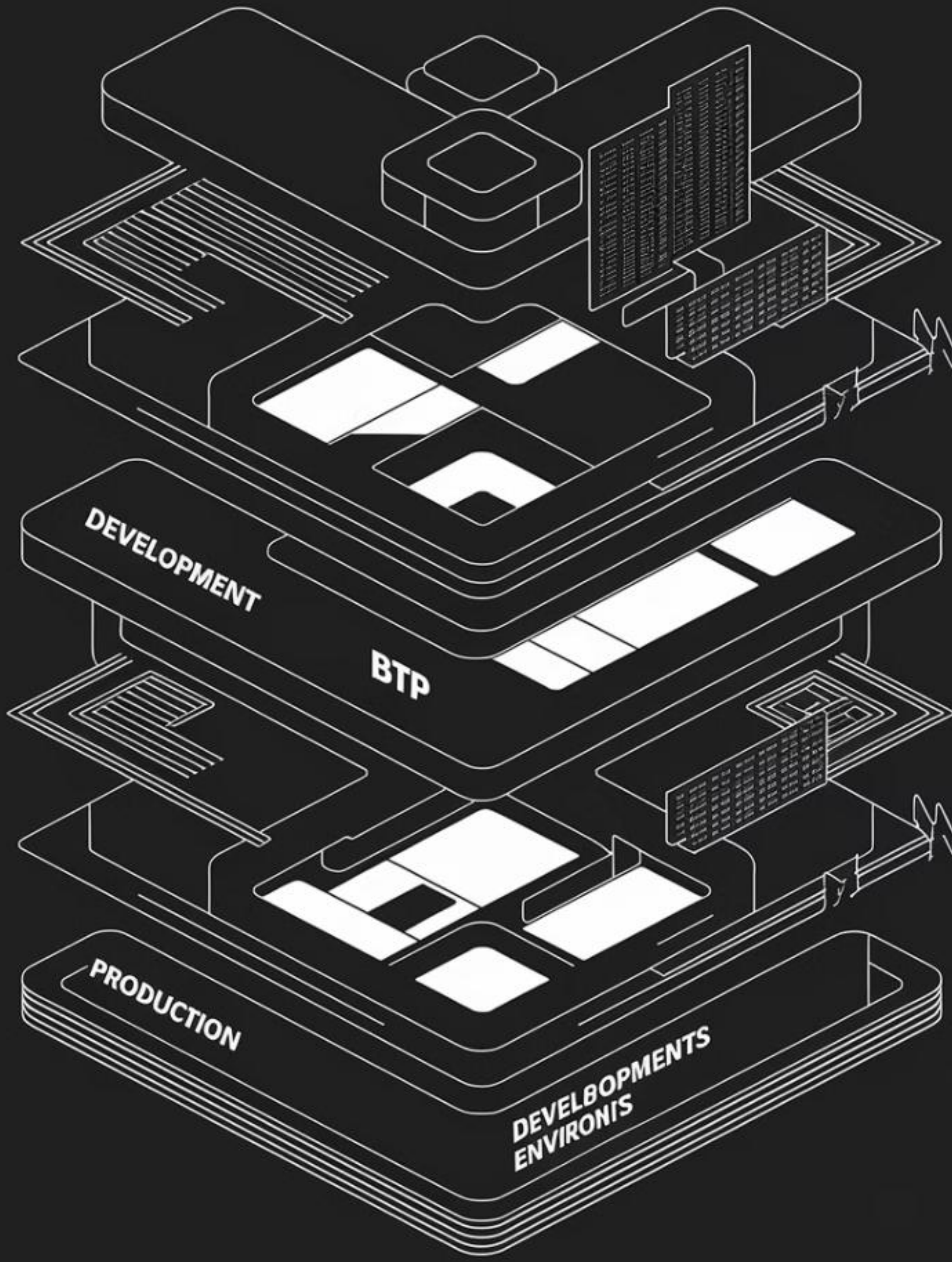
Cloud-native applications in SAP BTP should start quickly and handle termination signals properly. This is especially important in container orchestration environments like Kyma.

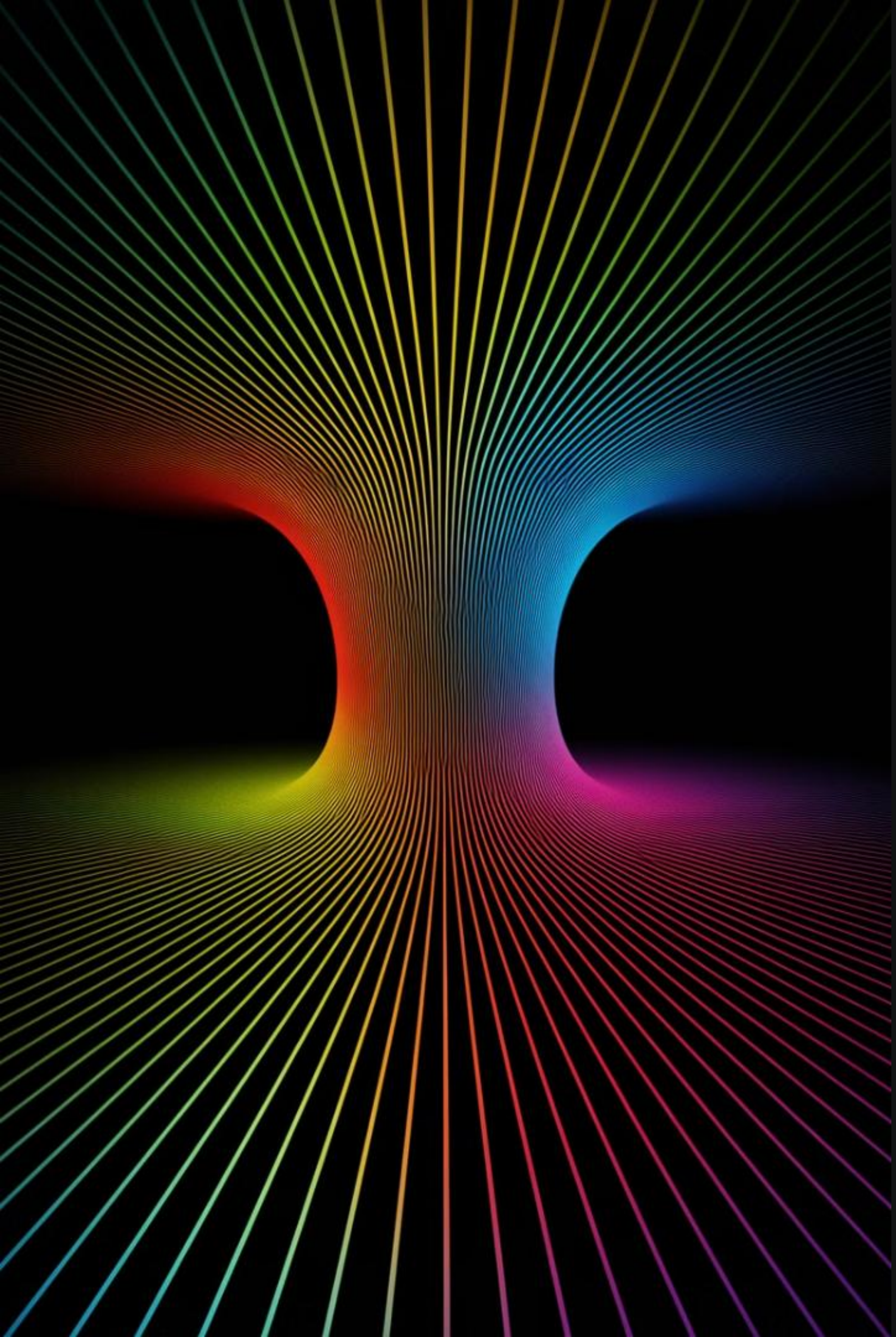

RAPID STARTUP
RAPIT PUFESSENTINGY
SMOOTH SHUTDOWN
SMOD SHUTDOWNT
SAP BUSISBIOLOS 8R PLATFORM
SNOR/ BUSINESS TECHNOLOGY
SMOOTH SHUTNOLOGY PLATFORM
SMOTH PUSIOIETERSIUS TECHNOOWLOGY

10. Dev/Prod Parity

Keep development, staging, and production as similar as possible.

SAP BTP supports creating consistent environments across development, testing, and production, reducing the "it works on my machine" problem.





11. Logs

Treat logs as event streams.

In SAP BTP, applications should write logs to stdout/stderr, which the platform then collects. The SAP BTP Cockpit provides access to these logs, or they can be forwarded to specialized logging services.

12. Admin Processes

Run admin/management tasks as one-off processes.

For SAP BTP applications, administrative tasks like database migrations should be packaged as one-off processes or jobs using the same codebase and configuration as the application.

Container-Based Solutions

Containers have become fundamental building blocks for cloud-native applications. They provide a consistent, isolated environment for applications across different infrastructure.

Containers in SAP BTP

SAP BTP supports container-based deployments through:

1. **Kyma Runtime**: A fully managed Kubernetes environment that allows deploying containerized applications directly on SAP BTP.
2. **Cloud Foundry Buildpacks**: While abstracting away containers, Cloud Foundry uses container technology behind the scenes to run applications.

Benefits of Containerization in SAP BTP

1 Consistency

The same container image runs identically across development, testing, and production environments.

2 Resource Efficiency

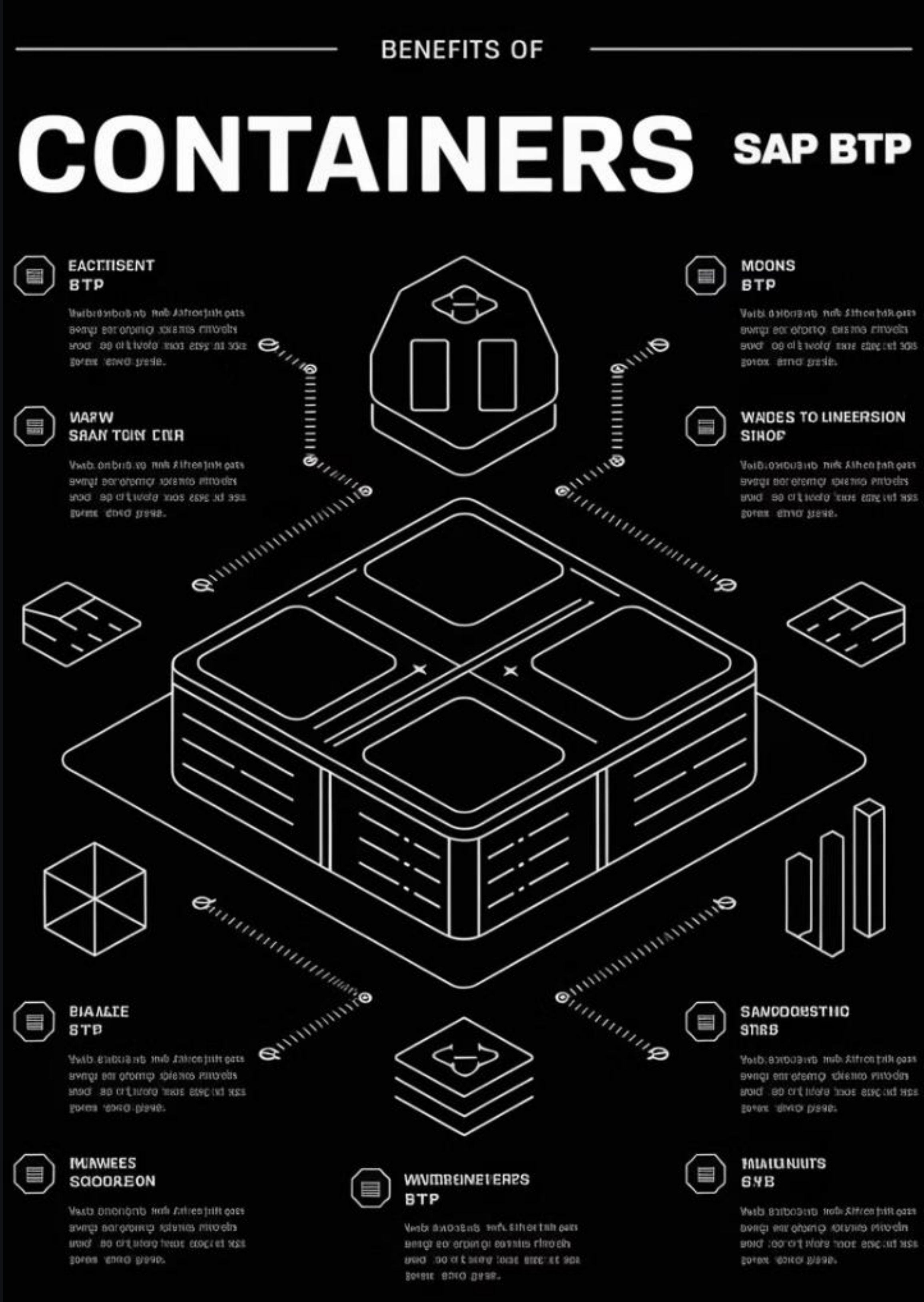
Containers share the host OS kernel, making them lightweight compared to virtual machines.

3 Isolation

Application dependencies and configurations are isolated from other applications.

4 Portability

Container images can be moved between different environments or cloud providers if needed.



Container Orchestration

For production workloads, container orchestration becomes critical. In SAP BTP, the Kyma runtime provides Kubernetes-based orchestration capabilities:

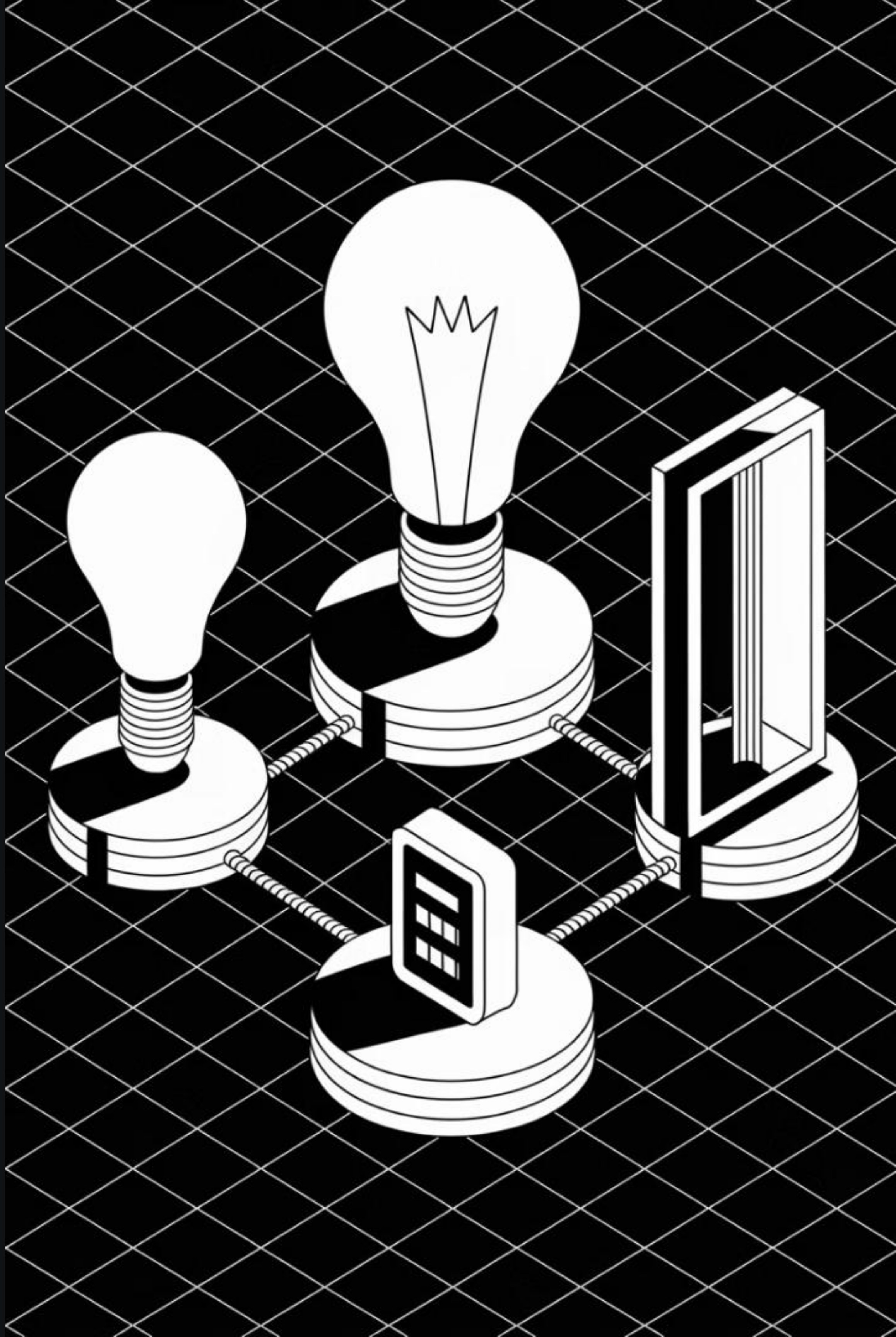
- **Automated deployment and scaling**: Deploy and scale containerized applications automatically.
- **Self-healing**: Automatically restart failed containers or reschedule them on healthy nodes.
- **Resource management**: Efficiently allocate and limit CPU and memory resources.
- **Service discovery and load balancing**: Route traffic to appropriate container instances.

Best Practices for Containers in SAP BTP

1. Keep container images small and focused on a single concern
2. Implement proper health checks for containers
3. Design applications to handle container restarts gracefully
4. Manage sensitive information using Kubernetes secrets
5. Implement proper logging for containerized applications

Service Discovery, Circuit Breakers, and API Gateways

As applications are decomposed into microservices, patterns for service communication become increasingly important.



Service Discovery

Service discovery solves the problem of finding and connecting to services in a dynamic cloud environment where instances come and go.

In SAP BTP, service discovery is implemented through:

1. **Cloud Foundry Routes**: In the Cloud Foundry environment, applications are registered with routes that remain stable even as instances change.
2. **Kubernetes Service Discovery**: In the Kyma runtime, Kubernetes Services provide a stable endpoint for a set of pods, implementing service discovery natively.
3. **Destination Service**: For cross-environment or external service communication, the SAP BTP Destination service provides a central repository of connection information.

Circuit Breakers

Circuit breakers prevent cascading failures by detecting when a service is unavailable and preventing further calls to it until it recovers.

Implementing Circuit Breakers in SAP BTP:

1. **Libraries and Frameworks**: Use libraries like Hystrix (Java), Resilience4j, or Polly (.NET) to implement circuit breakers in your application code.
2. **Istio Service Mesh**: In the Kyma runtime, Istio provides circuit breaking capabilities at the infrastructure level.
3. **SAP Cloud SDK**: The SAP Cloud SDK includes resilience features like circuit breakers for Java and JavaScript applications.

API Gateways

API gateways serve as the entry point for clients to access various microservices, providing cross-cutting concerns like authentication, rate limiting, and request routing.

API Gateway Options in SAP BTP:

1. **SAP API Management**: A comprehensive API management solution that provides API proxies, policy enforcement, analytics, and developer portals.
2. **API Rules in Kyma**: The Kyma runtime allows defining API rules to expose and secure microservices.
3. **Cloud Foundry Route Services**: In Cloud Foundry, route services can intercept and process requests before they reach the target application.

Example Architecture with These Patterns

Here's how these patterns might work together in an SAP BTP application:

1. External requests first hit an API gateway (SAP API Management)
2. The API gateway handles authentication, rate limiting, and forwards requests to the appropriate service
3. Service-to-service communication uses service discovery to locate dependencies
4. Circuit breakers protect services from cascading failures when dependencies are unavailable
5. All components are deployed as containers in the Kyma runtime or as applications in Cloud Foundry

Implementing Cloud-Native Architecture in SAP BTP

To implement cloud-native architecture effectively in SAP BTP, organizations should:

1

1. Choose the Right Runtime

- **Cloud Foundry**: Simpler deployment model, faster time to market
- **Kyma/Kubernetes**: More control, better for container-native applications

2

2. Adopt DevOps Practices

- Implement CI/CD pipelines using SAP BTP CI/CD services
- Automate testing at multiple levels (unit, integration, end-to-end)
- Practice infrastructure as code for environment provisioning

3

3. Implement Proper Monitoring

- Use SAP Cloud ALM for application monitoring
- Implement custom metrics for business-relevant indicators
- Set up alerts for proactive issue detection

4

4. Design for Failure

- Implement retry mechanisms with exponential backoff
- Use circuit breakers to prevent cascading failures
- Design services to degrade gracefully when dependencies fail

5

5. Secure by Design

- Implement zero-trust security model
- Use SAP BTP security services for authentication and authorization
- Scan container images for vulnerabilities

Conclusion

Cloud-native architecture represents a fundamental shift in how applications are built and operated on SAP BTP. By embracing the twelve-factor principles, container-based deployments, and modern communication patterns, organizations can create applications that are resilient, scalable, and maintainable.

SAP BTP provides all the necessary services and runtimes to implement cloud-native architecture effectively. Whether using Cloud Foundry for rapid application development or Kyma for container-orchestrated microservices, the platform supports modern cloud-native practices.

As cloud-native technology continues to evolve, SAP BTP will continue to incorporate new capabilities and best practices, ensuring that applications built on the platform can leverage the latest innovations in cloud computing.