Yandex.Cloud

Requirements and recommendations for building a PCI DSS infrastructure

Version 1.0

Yandex Cloud

Yandex.Cloud. Requirements and recommendations for building a PCI DSS infrastructure. Version 1.0.
This document is an integral part of the Yandex.Cloud technical documentation.

Notice about exclusive rights and confidential information

Contact information Yandex.Cloud LLC https://cloud.yandex.ru/

Phone: +7 495 739 7000
Email: cloud_docs@yandex-team.ru
Head office: 119021, Moscow, Russia, ul. Lva Tolstogo, 16

Revision history

Version	Revisions
1.0	Initial version of document

Table of contents

Introduction

Yandex.Cloud customers that use the platform's components for PCI DSS compliance shall use this document as a guideline for building an infrastructure that meets PCI DSS requirements.

This document is an integral part of the responsibility matrix. During the PCI DSS compliance audit, a QSA auditor (Qualified Security Assessor) shall assess and verify compliance with the requirements stipulated herein.

This document was prepared taking into account the following documentation:

- PCI DSS v3.2.1 — May 2018: https://www.pcisecuritystandards.org/documents/PCI_DSS_v3-2-1.pdf
- Information Supplement: PCI SSC Cloud Computing Guidelines: https://www.pcisecuritystandards.org/pdfs/PCI_SSC_Cloud_Guidelines_v3.pdf

Course of action to achieve PCI DSS compliance

To achieve PCI DSS compliance, the following actions must be done:

- Review the responsibility matrix between the Customer and Yandex.Cloud platform.
- Review this document.
- Build an infrastructure that processes payment card data (Cardholder Data Environment, CDE) based on this document and the responsibility matrix.
- Implement the PCI DSS requirements in the Customer's scope of responsibility and shared scope of responsibility.
- Select a QSA auditor and run the PCI DSS audit for the created infrastructure.

For any questions regarding the compliance of the Yandex.Cloud platform with the PCI DSS standard, please contact Yandex.Cloud support.

Scope of PCI DSS compliance for Yandex.Cloud components

Yandex.Cloud Platform meets all the PCI DSS 3.2.1 requirements for Level 1 Service Providers. Customers can use Yandex.Cloud services to build an infrastructure that meets PCI DSS requirements.

The scope of compliance includes the platform's following services:

Infrastructure & Network

- Yandex Compute Cloud
- Yandex Object Storage
- Yandex Virtual Private Cloud
- Yandex Network Load Balancer
- Yandex Cloud Interconnect
- Yandex Instance Groups
- Yandex API Gateway

Containers

- Yandex Container Registry
- Yandex Managed Service for Kubernetes®

Security

- Yandex Identity and Access Management
- Yandex Certificate Manager
- Yandex Key Management Service

Data Platform

- Yandex Managed Service for PostgreSQL
- Yandex Managed Service for ClickHouse
- Yandex Managed Service for MongoDB
- Yandex Managed Service for MySQL®
- Yandex Managed Service for Redis™
- Yandex Data Proc
- Yandex Managed Service for Apache Kafka®
- Yandex Managed Service for Elasticsearch
- Yandex Database

Serverless

- Yandex Message Queue
- Yandex Cloud Functions
- Yandex API Gateway
- Yandex Database

- Yandex Object Storage

Operations

- Yandex Resource Manager

Data classification in Yandex.Cloud

In this document, PCI DSS data refers to the data whose processing and storage necessitates compliance with the PCI DSS standard. Payment card data (Account Data)	
Cardholder Data	Sensitive Authentication Data
• Card number (PAN) • Cardholder name • Card expiration date • Service code	• Full track data: data from the magnetic stripe or chip • CAV2/CVC2/CVV2/CID • PINs and (or) PIN blocks

PCI DSS infrastructure means a set of Yandex.Cloud services that the Customer uses to store, transmit, and process PCI DSS data.

Secret data (or secrets) mean the private cryptographic keys, encryption keys, authentication tokens, and so on: i.e., datasets, which the possession of enables one to obtain or facilitate access to PCI DSS data or any other sensitive data.

Meeting requirements by using Yandex Identity and Access Management

Managing privileged users

Yandex.Cloud privileged users shall include accounts with the following roles:

- resource-manager.clouds.owner
- billing.accounts.owner
- admin assigned for the entire cloud
- admin assigned for the folder
- admin assigned for a billing account

The billing.accounts.owner role is automatically issued when creating a billing account and can't be reassigned to another user. The role lets you perform any action with the billing account.

The billing.accounts.owner role can only be granted to a Yandex.Passport account.

An account with the billing.accounts.owner role is used when setting up payment methods and adding clouds. For the duration that this account is actively used, be sure to enable two-factor authentication (2FA) in Yandex.Passport. After that, if you don't use the bank card payment method (available only for this role), we recommend that you set a strong password for this account, disable 2FA, and refrain from using this account unnecessarily. We recommend that you change the password each time you use it.

To manage a billing account, we recommend that you assign the admin or editor role for the billing account to a dedicated employee with a federated account. To view billing

data, we recommend that you assign the viewer role for the billing account to your dedicated employee with a federated account.

The resource-manager.clouds.owner role is assigned automatically when you create a cloud. A user with this role can perform any operation with the cloud or its resources and grant cloud access to other users: assign roles and revoke them.

We recommend that you assign the resource-manager.clouds.owner role to one or more of your employees with a federated account, set a strong password for the Yandex.Passport account that was used to create the cloud, and use it only when absolutely necessary (for example, if the federated access fails). Be sure to fully protect your federated account with the resource manager.clouds.owner role:

- Two-factor authentication must be enabled.
- Authentication from devices beyond the company's control must be disabled.
- Login attempt monitoring must be configured with the alert thresholds set.

We recommend that you assign to your federated accounts the admin roles for the cloud, folders, and billing accounts, minimize the number of accounts with such roles, and regularly review the expedience of such roles for the assigned accounts.

Two-factor authentication

The PCI DSS standard requires two-factor authentication for accessing the infrastructure. This requires 2FA for accessing the Yandex.Cloud console. To meet this requirement, the Customer must use an identity provider that supports 2FA and set up a SAML-compliant federation for accessing the Yandex.Cloud console.

Using a resource model

For developing an access model for the PCI DSS infrastructure, we recommend the following approach:

- All the resources that must be PCI DSS-compliant must be placed in a separate cloud.
- We recommend that you host the resource groups requiring different administrative permissions in different folders (DMZ, CDE, security, backoffice, etc.).
- We recommend that you host the shared resources (such as network and security groups) in a separate shared resource folder.

Using service accounts

When using service accounts, we recommend that you:

- Assign a service account to a VM instance and get a token using the metadata service.

- Set up a local firewall in the VM instance so that only the necessary processes and system users have access to the metadata service (IP address: 169.254.169.254).
- Store the service account keys and manage them according to the PCI DSS requirements.
- Follow the principle of minimum privileges and assign to the service account only those roles that are needed to run the application.
- Follow the principle of minimum privileges for the service account as a resource: assign the roles for using and managing your service accounts to a minimum number of users who truly require such roles.

Minimum privileges and security policy

We don't recommend that you use the primitive roles: admin, editor, member, viewer. To ensure a higher granularity of access control and implementation of the principle of minimum privileges, you need to use service roles when building your PCI DSS infrastructure.

Meeting the requirements using Yandex Virtual Private Cloud and Yandex Cloud Interconnect

Encryption of transmitted data

To meet the PCI DSS requirements when using Yandex VPC and Yandex Cloud Interconnect, enable encryption in transit at the application level, for example, by using TLS 1.2 or higher. In this case, your settings and algorithms must be PCI DSS-compliant.

Yandex.Cloud API supports PCI DSS-compatible sets of algorithms (cipher suits) and TLS versions. When using the Yandex.Cloud API, ensure that the TLS client can't connect via insecure TLS protocols (versions below 1.2) or ensure that no insecure protocols will be used while establishing connections. For example, by using the gRPC interfaces of Yandex.Cloud, you can enforce TLS 1.2 or higher. That's because gRPC is based on HTTP/2 where TLS 1.2 is the minimum supported TLS version. Legacy TLS protocols will gradually be discontinued in Yandex.Cloud services.

Delivery of traffic to the app and network segmentation

To protect virtual machines at the VPC level and allocate DMZ and other network segments, we recommend that you use security groups.

To deliver traffic to an application within the PCI DSS infrastructure, we recommend that you use a network load balancer to route your traffic through the selected ports only.

We recommend that you use the network load balancer together with security groups to limit the list of IP addresses that have access to the application.

Outbound internet access

To enable outbound internet access, use static IP addresses that can be added to the exceptions list of the receiving party's firewall.

We don't recommend that you use a NAT gateway, since the NAT gateway's IP address might be used by multiple users at the same time. This feature must be taken into account when modeling threats for your Yandex.Cloud-based PCI DSS infrastructure.

When using dynamic public IP addresses and static IP addresses, make sure that the security groups with the PCI DSS-compliant sets of rules are applied to the resources.

Flow Logs and network IDS/IPS

It is the Customer's responsibility to write Flow Logs and use the IDS/IPS systems in VPC. We recommend that you use popular IDS (for example, Suricata or Snort).

To manage traffic flows and route them through IDS, you can use static VPC routes.

Enabling administrative access to the PCI DSS infrastructure

To enable access to the PCI DSS infrastructure over SSH, we recommend that you create a Bastion VM instance or VPN gateway. Online access to the Bastion VM instance or a VPN gateway must be restricted by using security groups. See the example of creating a secure VPN gateway in the documentation.

For better control of administrative actions, we recommend that you use PAM (Privileged Access Management) solutions with administrator sessions recording (for example, Teleport).

For SSH and VPN access, we recommend that you avoid using passwords and use public keys, X.509 certificates, and SSH certificates instead. When setting up SSH for your virtual machines, we recommend that you use the SSH certificates (for the SSH host as well).

Meeting the requirements when using Yandex Compute Cloud

Encryption of stored data

When using Yandex Compute Cloud, keep in mind the following:

- Requirement 3.2 of PCI DSS prohibits storing critical authentication data after authorization.
- If disk encryption is used on a virtual machine in order to meet PCI DSS Requirement 3.4, this is the sole responsibility of the Customer. This applies to any type of the platform's disks, including non-replicated disks (NRD).

- It is prohibited to store the secrets used for data access as clear text on the disks of virtual machines. To learn how to correctly work with secrets, see Meeting the requirements when using Yandex Key Management Service.

Important: Requirement 8.2.1 of PCI DSS prohibits storing the secrets (API keys, certificates) needed to run the PCI DSS applications on virtual machine disks and in virtual machine images as clear text.

To enable encryption, you can attach an additional disk to the virtual machine with Full Disk Encryption enabled and host your application files on it.

Using virtual machine metadata

It is prohibited to write PCI DSS data and secrets as clear text both in metadata (user-data, etc.) and in the virtual machine names and descriptions.

Recommendations for storing secrets in the KMS service are given in Meeting the requirements when using Yandex Key Management Service.

Using a serial console

When running a serial console:

- Make sure that PCI DSS data is not output to the serial console.
- For improved security, we recommend that you disable interactive access to the serial console. Risks from using a serial console are listed in the documentation.
- If you enable SSH access to the serial console, make sure that both the credentials management and password used for local login to the operating system are PCI DSS-compliant.

Recommendations for virtual machine deployment

When deploying virtual machines in a PCI DSS infrastructure, we recommend that you:

- Prepare a virtual machine image (for example, using packer) with PCI DSS-compliant system settings.
- Use this image to create a virtual machine or instance group.
- Look up the virtual machine's details to check that it was created using this very image.

Dedicated hosts and side-channel attacks

To ensure the best protection against CPU side-channel attacks (for example Spectre or Meltdown):

- Use full-core virtual machines, i.e., instances with a CPU share of 100%.
- Use virtual machines with an even number of cores (2 cores, 4 cores, etc.).

- Install updates for your operating system and kernel that ensure side-channel attack protection (for example, Kernel page-table isolation for Linux, applications built using Retpoline).

We recommend that you use dedicated hosts for the most security-critical resources.

Meeting the requirements when using Yandex Object Storage

Encryption of stored data

When using Yandex Object Storage, be sure that PCI DSS data is encrypted. Use any of the following approaches for encrypting the data:

- Encryption of the Object Storage bucket using KMS keys (Server Side Encryption) is the recommended approach. This encryption method protects against the accidental or intentional publication of bucket contents on the internet.
- Object Storage integration with KMS for Client-Side Encryption.
- Third-party encryption libraries at the application level prior to sending the data to Object Storage.

When using third-party libraries for data encryption and your own methods of key management, be sure that your operating model, algorithms, and key lengths are PCI DSS-compliant.

Encryption of transmitted data

When using Object Storage, be sure that support for TLS protocols below version 1.2 is disabled at the client level.

Use the bucket policy `aws:securetransport` to ensure that running without TLS is disabled for the bucket.

Bucket ACLs and access policies

For improved data protection, we recommend that you install strong ACLs for buckets and use bucket policies, for example, `aws:sourceip`, in order to restrict the IP addresses that can access the bucket.

We don't recommend that you assign roles for entire folders in Object Storage. We recommend that you minimize the permission scope and only assign it for certain buckets or objects in Object Storage.

Meeting the requirements when using Yandex Key Management Service

Encryption key length

We recommend that you use Key Management Service (KMS) to encrypt data. KMS uses the AES-GCM encryption mode. The Customer can select the key length: 128/192/256 and set up the preferred key rotation period.

Authentication and authorization in the KMS service

To access the KMS service, use an IAM token (see Meeting the requirements when using Yandex Identity and Access Management). We recommend that you get the IAM token for your service account using the mechanism of assigning a service account to your virtual machine.

We recommend that you grant granular permissions for specific keys in the KMS service to your users and service accounts.

Recommended cryptographic libraries

For client-side encryption, we recommend that you use the following libraries:

- AWS Encryption SDK and its KMS integration.
- Google Tink and its KMS integration.
- Yandex.Cloud SDK with any other PCI DSS-compatible cryptographic library.

Storing secrets using KMS

To store secrets (access tokens, API keys, etc.) we recommend the following procedure:

1. Prepare a file with your secrets and encrypt it using the KMS key on behalf of the user.
2. Revoke the user's permission for operations with this key.
3. Create a service account for the application and grant it permission to use an encryption key (the kms.keys.encrypterDecrypter role).
4. When the application starts, get your IAM token from the metadata service and decrypt the secrets in KMS.
5. Use the secrets in the application's memory, save them to a RAM disk (tmpfs), or use the operating system's security mechanism (for example, Linux Kernel Keyrings).

Using secrets and keys in Terraform

When using KMS with Terraform, keep in mind that the secrets in this case are stored as clear text in the configuration files and will also end up in "state" as clear text. Therefore, when using Terraform, be sure to protect both the configurations and "state" based on the PCI DSS requirements.

When running KMS in Terraform, we recommend that you use the lifecycle block to prevent key deletion and possible data loss.

KMS key rotation

Many Yandex.Cloud services only process data without storing it. To improve the security of your PCI DSS infrastructure, we recommend that you categorize your keys into two groups:

1. Keys for services that process PCI DSS data (for example, Yandex Message Queue, Yandex Functions), but don't store such data.
2. Keys for the services that store PCI DSS data (for example, the Data Platform services).

For the first group of keys, we recommend that you set up automatic key rotation with a rotation period slightly longer than the data processing period in these services. When the rotation period expires, the old key versions must be deleted. In the case of automatic rotation and the deletion of old keys, already processed PCI DSS data can't be restored and decrypted.

Read more about key rotation in the KMS documentation.

For data storage services, we recommend that you use either "manual" rotation procedures or automatic key rotation, depending on your internal procedures for processing PCI DSS data.

Important: A secure value for the AES-GCM mode is encryption using 2^{32} blocks. Having reached this number of encrypted blocks you need to create a new DEK. For more information about the AES-GCM operating mode, see the NIST materials.

Important: Before deleting versions of your keys, make sure that you won't lose any data as a result of the operation. You can protect a key against deletion by setting the parameter deletionProtection. For more information about the procedure for deleting keys in KMS, see the documentation.

Meeting the requirements when using managed database services

Data protection

It is prohibited to use PCI DSS data or secrets (API keys, encryption keys, etc.) as cluster names, cluster descriptions, database names, usernames, and so on.

Password policy

The password must:

- Include numbers, uppercase letters, lowercase letters, and special characters.
- Have a length of at least 8 characters.

Managing accounts

Be sure to create and rotate account passwords in accordance with the PCI DSS requirements.

Be sure to regularly check the number of your DBMS accounts and their privileges.

Data encryption

Be sure to encrypt PCI DSS data at the application level (for example, using the KMS service) before writing this data to the service's databases.

Managed databases use TLS 1.2 and higher.

Network access to databases

The Customer must disable other services (such as Data Lens, the management console, etc.) from accessing the databases that host PCI DSS data. You can do it in the cluster settings or when creating the cluster.

It is prohibited to provide online access to the databases that include payment card data or other sensitive data. In any cases where such access is required, be sure to set up security groups to grant access to the service from only trusted IP addresses.

In addition to restricting access at the VPC level, we recommend that you use security groups that allow DBMS connections from only certain private (RFC 1389) addresses.

Meeting the requirements when using Yandex Data Proc

In Yandex Data Proc, the Customer is responsible for the contents of virtual machines and their PCI DSS compliance.

Yandex Data Proc uses Yandex Object Storage to store processed data. Therefore, to be PCI DSS compliant, you must follow the recommendations for using Yandex Object Storage.

Meeting the requirements when using Yandex Message Queue

If the YMQ service is used for transmitting PCI DSS data or secrets (encryption keys, API keys, etc.), be sure to encrypt this data at the application level before sending them to YMQ. For the KMS key, we recommend that you set up a rotation period greater than or equal to the maximum YMQ message processing time.

Meeting the requirements when using Yandex Cloud Functions and Yandex API Gateway

Data processing

When using Yandex Cloud Functions, it is prohibited:

- To store PCI DSS data, as well as secrets (certificates, API keys, etc.) as clear text in the Cloud Functions code. A proper way for delivering secret data is described in the section Meeting the requirements when using Yandex Key Management Service.
- To use PCI DSS data as function names, descriptions, and labels.
- To write PCI DSS data as clear text to the Cloud Logs (log groups). If you want to write PCI DSS data, be sure to encrypt it using KMS or any other PCI DSS-compliant method .

Using a service account

To get an IAM token while running a function, create the function's version for the service account and use the function to obtain an IAM token as described in the documentation.

Function access control

In cases where the use of public functions is not explicitly required, we recommend that you use private functions.

Side-channel attacks in Cloud Functions

Hosts and hypervisors running Cloud Functions contain all the applicable updates for side-channel attack protection. However, keep in mind that different clients' functions are not isolated by cores. Thus, there technically exists an attack surface between one user's function and another user's function.

Yandex.Cloud security experts believe that side-channel attacks are unlikely in the context of functions, but this risk must be accounted for in the overall threats and risk analysis model employed by the PCI DSS infrastructure.

Specifics of time synchronization in functions

The Cloud Functions service does not guarantee time synchronization prior to or during execution of requests by functions. To generate a function log with exact timestamps on the Cloud Functions side, output the log to stdout.

The Customer can also independently accept function execution logs and label them with a timestamp on the receiving side. In this case, the timestamp is taken from the time source synced with Yandex.Cloud.

Meeting the requirements when using Managed Service for Kubernetes

Responsibility matrix

All the actions inside the Kubernetes node shall be the Customer's responsibility. The Customer is responsible for the security of nodes and their proper configuration in accordance with PCI DSS requirements.

Yandex.Cloud is responsible for the security of the Kubernetes API.

The Customer is responsible for the proper selection of the security settings for the Managed Service for Kubernetes and selection of the update channel and/or schedule.

We don't recommend that you provide access to the Kubernetes API from untrusted networks, for example, the internet. If this is necessary, set up security groups.

Data processing

When running Managed Service for Kubernetes, it is prohibited:

- To use PCI DSS data in cluster names, descriptions, IDs, labels, etc.
- To write PCI DSS data to pod manifests.
- To write PCI DSS data to etcd as clear text.
- To write PCI DSS data to Managed Service for Kubernetes logs.

Storing secrets

For the management of secrets, we recommend that you use the built-in Kubernetes functionality. However, be sure to use Kubernetes-to-KMS integration in this case.

It is prohibited to store secrets (API keys, secret keys of certificates, etc.) in Kubernetes pod manifests as clear text.

Authentication and authorization, account management

We recommend that you use IAM integration for authentication within clusters. To access the cluster, we recommend that you use the service roles: k8s.admin or k8s.user.

For the cluster's service account, we recommend that you use the role k8s.cluster.sa.

For a node group service account used for authentication in Container Registry, we don't recommend assigning any roles at the folder level. Instead, we recommend that you grant object permissions at the registry level (see Meeting the requirements when using Yandex Container Registry and Yandex Container Solution).

Encryption of stored data

If you need to encrypt your stored data, you can use:

- KMS to encrypt the data at the application level.
- Your custom method for data encryption. However, your key security and key management procedures must be PCI DSS-compliant.

Security monitoring and enhanced cluster protection

We recommend that you use additional tools for monitoring the security of Managed Service for Kubernetes. For example:

- Falco or kube-query as HIDS.

- Kubernetes' built-in support for AppArmor and Seccomp.
- Istio, for service-to-service authentication.
- ClamAV, for anti-virus protection.

You can also integrate SIEM systems and audit backends in Kubernetes.

To enhance the cluster's network security, you must create a cluster with enabled network policies and ensure pod isolation using network policies.

Load sharing between nodes

Data loads with different security contexts (i.e., different severities of data processed) must be processed on different Kubernetes nodes.

For load sharing within a cluster, use different node groups with different settings: node labels and node taints together.

Installing updates

Managed Service for Kubernetes releases new updates regularly. To comply with PCI DSS, you must:

- Select a relevant update channel and enable automatic or manual installation of updates immediately after publication in the selected channel.
- Check that the update settings are PCI DSS-compliant.
- Use one of the three latest Kubernetes versions, since updates (including security updates) are only released for those versions.

Using persistent volumes

When using persistent volumes, it is prohibited to write PCI DSS data to them as clear text. Make sure to encrypt PCI DSS data when storing data in persistent volumes (encryption at rest).

Backups

We recommend that you set up backups for Managed Service for Kubernetes by following the documentation.

When storing backups in Object Storage, follow the recommendations from the section Meeting the requirements when using Yandex Object Storage (for example, use the built-in bucket encryption features).

Meeting the requirements when using Yandex Database

Operations with data

It is prohibited to use PCI DSS data as the names of databases, tables, columns, directories, etc.

It is prohibited to send PCI DSS data to YDB (dedicated and serverless) as clear text. Prior to sending data, be sure to encrypt the data at the application level. For this you can use the KMS service or any other PCI DSS-compliant method.

For data where the storage period is known in advance, we recommend that you configure the Time To Live option.

SQL injection protection

When working with the database, use parameterized prepared statements to protect against SQL injection. However, if the application uses dynamic generation of query templates, you must prevent the injection of untrusted user input into the SQL query template.

Network access

When accessing the database in Dedicated mode, we recommend that you use it inside VPC, disabling public access to it from the internet.

In Serverless mode, the database can be accessed from the internet. You must therefore take this into account when modeling threats to your PCI DSS infrastructure. When setting up database permissions, use the principle of minimum privileges.

Backups

When creating on-demand backups, be sure that the backup data is protected according to PCI DSS.

When creating backups on demand in Object Storage, follow the recommendations from the section Meeting the requirements when using Yandex Object Storage (for example, use the built-in bucket encryption feature).

Meeting the requirements when used for Yandex Container Registry and Yandex Container Solution

Authentication

For authentication in Container Registry, use the integrated authentication features based on IAM tokens. When you use Container Solution or Managed Service for Kubernetes, we recommend that you use the integrated authentication features.

For accessing the registry, we recommend that you use service roles.

For accessing Container Registry, we recommend that you use object-based access directly inside the registry.

Data protection

When using Container Registry and Container Solution, it is prohibited to store PCI DSS data, as well as secrets (private keys, API keys, encryption keys, etc.), in container images, in the YAML configurations of docker-compose, in Dockerfiles, and so on.

It is prohibited to write PCI DSS data and secrets (private keys, API keys, etc.) to container logs.

It is prohibited to use PCI DSS data and secrets as names and descriptions for registries, repositories, and other entities of the service.

Features for vulnerability management

In your vulnerability management processes, we recommend that you use Container Registry's vulnerability scanner for packages and any related scanners for images.

In Container Solution, the Customer runs a virtual machine with containers. Therefore, the Customer is responsible for the Container Solution virtual machine's compliance with PCI DSS.

Additional security features

We do not recommend that you use privileged containers to run loads that process untrusted user input. Privileged containers must be used for the purposes of administering virtual machines or other containers.

We recommend that you use delete policies for automatically deleting outdated container images.

Meeting the requirements when using Yandex Certificate Manager

Certificate Manager lets you manage gateway certificates for Yandex API Gateway and sites and buckets in Yandex Object Storage. We recommend that you use Certificate Manager to obtain your certificates and rotate them automatically.

When using TLS in your application, we recommend that you limit the list of your trusted root certificate authorities (root CA). When using certificate pinning, keep in mind that Let's Encrypt certificates are valid for 90 days.

Security rules when contacting support

When contacting Yandex.Cloud support, it is prohibited:

- To send PCI DSS data via email, feedback forms, or any other methods.

- To send your credentials in messages and disclose them to support representatives: passwords, IAM tokens, service account keys, application secrets, etc.

Security workflow for PCI DSS

Getting system logs and actions in case of incidents

If you detect any information security incidents, please contact Yandex.Cloud support.

To request additional logs, follow the data request procedure.

Time synchronization

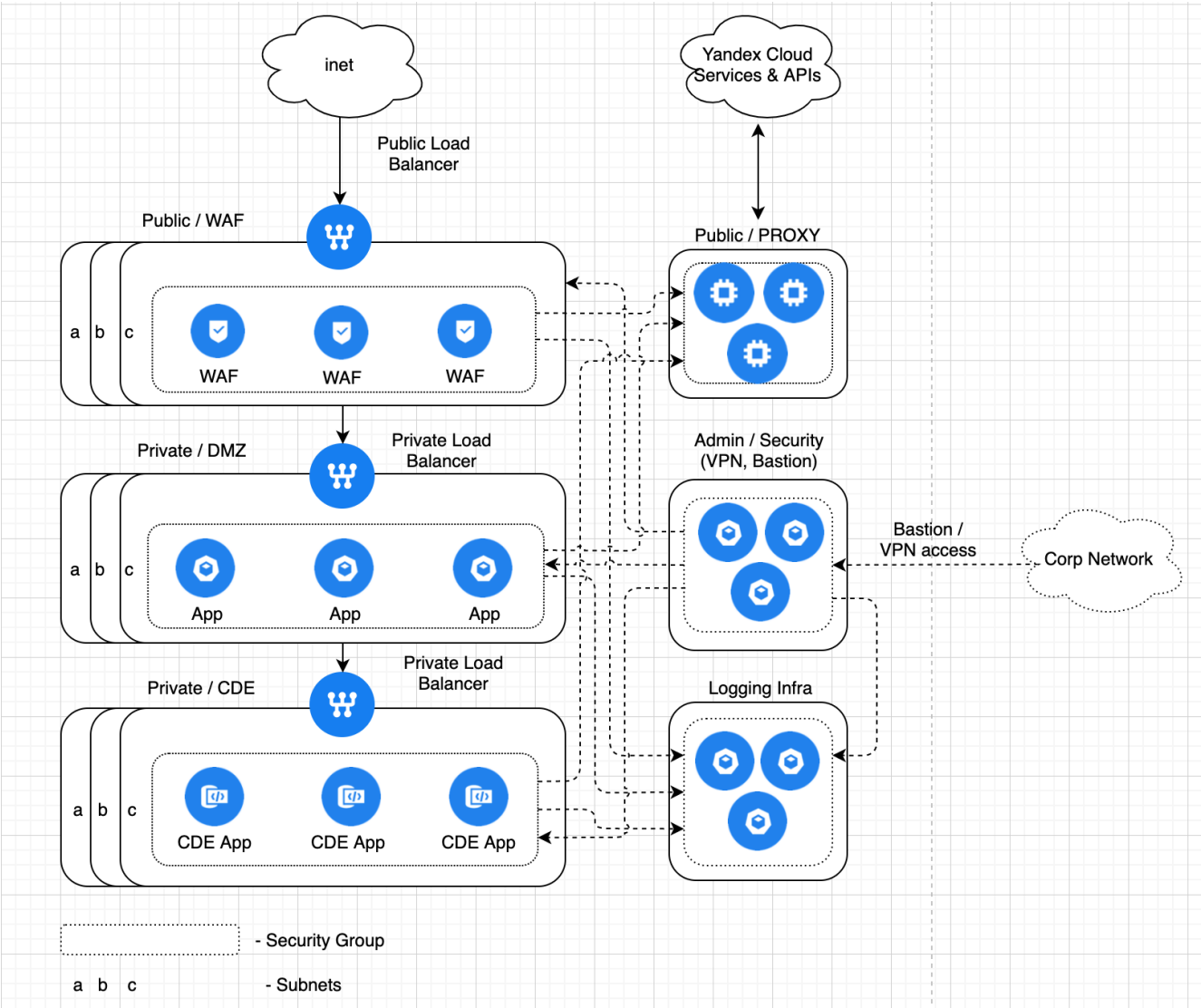
When building your PCI DSS infrastructure on Yandex.Cloud, use the time synchronization settings from the documentation.

Running infrastructure scans

When running penetration tests for the PCI DSS infrastructure deployed on Yandex.Cloud, follow the rules for external security scans.

Example of an application architecture with CDE in Yandex.Cloud

An example of a typical PCI DSS-compliant architecture is shown in the diagram. The infrastructure is split into 6 different security contexts, with segmentation between the contexts enabled by security groups.



Dotted arrows indicate the rules that enable interaction between the security groups (that coincide with the security contexts in this case).

All the components of the application are hosted in three different availability zones to ensure fault-tolerance.

Access to the Yandex.Cloud API is implemented using the Proxy layer that controls access parameters. Depending on the implementation, it can run MITM prevention and traffic introspection.

The infrastructure can be accessed using a Bastion host or VPN gateway.