**Instana POV Scoping: Questionnaire**

This questionnaire gathers business and technical information about the initiatives, goals, applications, frameworks, and environments in which we will conduct the Instana POV. Please *review* prior to our scoping session as we will walk through this together on our POV scoping call.

Any questions, please contact your assigned Instana Solutions Architect.

Thank you.

TABLE OF CONTENTS:

[**General Business and Background Information**](#_Toc92462081)

[**Monitoring Problem Statements**](#_Toc92462082)

[**Current Tooling**](#_Toc92462083)

[**POV Application Info**](#_Toc92462084)

[**Infrastructure**](#_Toc92462085)

[**Java Runtimes**](#_Toc92462086)

[**Python Runtimes**](#_Toc92462087)

[**Ruby Runtimes**](#_Toc92462088)

[**Node Runtimes**](#_Toc92462089)

[**.NET Runtime**](#_Toc92462090)

[**POV Planning, Network and Security Requirements**](#_Toc92462091)

## **General Business and Background Information**

Company Name:

Company / Department Description:

Customer Main Contact Name:

Customer Main Contact Email:

Instana Solution Architect Name:

Instana Solution Architect Email:

Planned POV Start Date

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Describe your key IT/Ops and/or software development initiatives.

Where are you in your cloud first or cloud native journey?

* What % of your IT environment is traditional vs. cloud (SaaS)?
* What is your cloud native platform of choice?

Based on your technology initiatives, describe where you are in your journey transforming your organization to support the adoption of new technologies (e.g., waterfall to agile to CI/CD, dev/ops/SRE):

What compelling events are you facing (e.g., downtime/MTTR issues, major software deliverables/dates, compliance, etc.)?

Describe the most critical business capabilities that your monitoring environment supports today. Why are they critical? How do they impact the overall business?

* Who are the primary consumers/users of your monitoring solutions?
* How many people are in these teams/groups in total?
* How do your monitoring solutions impact these teams/groups?

From a monitoring perspective, what are the biggest challenges you would like to try to address with a new monitoring solution? Let’s articulate these as specific problem statements in the next section(s).

## **Monitoring Problem Statements**

**Problem Statement 1**

What is the problem you are trying to solve?

How are you currently addressing it?

How would you validate the envisioned solution (recommended success criteria)?

What type of measurements would indicate success (for example number of deployments per day vs. number of production alerts per day, reduction in MTTR, etc.)?

What SLAs (Service-level agreement) does this problem affect?

**Problem Statement 2**

What is the problem you are trying to solve?

How are you currently addressing it?

How would you validate the envisioned solution (recommended success criteria)?

What type of measurements would indicate success (for example number of deployments per day vs. number of production alerts per day, reduction in MTTR, etc.)?

What SLAs does this problem affect?

**Problem Statement 3**

What is the problem you are trying to solve?

How are you currently addressing it?

How would you validate the envisioned solution (success criteria)?

What type of measurements would indicate success (for example number of deployments per day vs. number of production alerts per day, reduction in MTTR, etc.)?

What SLAs does this problem affect?

## **Current Tooling**

What infrastructure monitoring tools do you use?

What tracing tools do you use?

What log analysis tools do you use?

What dashboarding tools do you use?

What synthetic monitoring tools do you use?

What end-user monitoring tools do you use?

What load testing tools do you use?

What CI / CD tools do you use?

Deployment interval (daily, weekly, monthly, etc.)?

What ITSM solution(s) do you use (e.g., HPOV, BMC, SNOW, PagerDuty)?

## **POV Application Info**

Application Name to be POV’d

What is the business function of the application?

Is this a mission critical application (y/n)?

Who is impacted by this application not running (or not running well)?

How would you describe the architecture of the application?

* Monolithic
* SOA
* Microservices
* C/S
* Other

Comments

Broadly speaking, how does one interact with the application?

* Public API
* Private (internal only) API
* Web Application
* Mobile Application
* Other

Where is the application deployed (on-prem, cloud, hybrid etc.)?

Approximately, how many hosts will be monitored (or nodes if K8s or OpenShift)?

If applicable, approximately how many containers will be monitored (also avg container density)?

If applicable, how many public cloud managed services will be monitored?

In what environment(s) will we conduct the POV (dev, stage, prod, etc.)?

## **Infrastructure**

**Infrastructure - Operating System**

 Ubuntu Linux 14.04 (trusty)

 Ubuntu Linux 16.04 (xenial)

 Ubuntu Linux 18.04 (beaver)

 CentOS 6

 CentOS 7

 Debian 9 (stretch)

 SUSE (SLES) 12

 Redhat Enterprise Linux (RHEL) 6

 Redhat Enterprise Linux (RHEL) 7

 Amazon Linux 1

 Amazon Linux 2

 Windows

 AIX

 HP-UX

 Solaris

 z/OS

 Other:

**Infrastructure - Container**

 Docker

 Containerd

 LXC

 RCKT

 CRI-O

 Garden

 Other:

**Infrastructure - Orchestration**

 Kubernetes >=1.9.x

 OpenShift 3.7,3.9,3.10,3.11

 Amazon Elastic Container Service for Kubernetes (EKS): 1.10.x >=1.10.x

 Azure Kubernetes Service (AKS): 1.10.x >=1.10.x

 Google Kubernetes Engine (GKE): 1.10.x, 1.11.x, 1.12.x >=1.10.x

 Pivotal Cloud Foundry >=2.3

 Other:

**Infrastructure - Cloud Services**

 AWS - Beanstalk

 AWS - DynamoDB

 AWS - EC2

 AWS - EBS

 AWS - ElastiCache

 AWS - EMR

 AWS - ELB

 AWS - Kinesis

 AWS - Lambda

 AWS - MQ

 AWS - RDS

 AWS - SQS

 AWS - S3

 AWS - XRay (technology preview)

 AWS - Elasticsearch

 Azure - API Management

 Azure - App Service

 Azure - CosmosDB

 Azure - Redis Cache

 Azure - SQL Database

 Azure - SQL Elastic Pool

 Azure - SQL Server

 Azure - Storage

 Google - Cloud SQL

 Google - Compute Engine

 Other:

**Infrastructure - Service Mesh**

 Istio

 Linkerd

 Linkerd2

 Envoy

 Other

**Infrastructure – Messaging/Message Brokers**

 Kafka

 Mule ESB

 IBM MQ

 Other:

**Infrastructure - Web Server**

 Apache httpd

 IBM IHS

 Nginx

 Other:

**Infrastructure - Databases**

 MySQL

 Oracle

 DB2

 Microsoft SQL Server

 MariaDB

 Cassandra

 Clickhouse

 CockroachDB

 Couchbase

 Elasticsearch

 Memcached

 MongoDB

 PostgreSQL

 Redis

 Other:

## 

## **Java Runtimes**

 JVM runtimes v6 to v17

**Java JVM Type**

 Amazon Corretto

 Azul Zulu

 IBM J9

 OpenJDK

 Oracle Hotspot

 SAP JVM

 Sun Hotspot

 BEA JRockit

 Eclipse OpenJ9

 Other:

**Java HTTP Clients**

 Akka HTTP >= 2.4, >= 10.0

 Apache Async HttpClient >= 4.0

 Apache CXF >= 3.0

 Apache HttpClient >= 3.0, >= 4.3

 AsyncHttpClient >= 2.0.9, >= 2.1.0

 Axis >= 1.3

 Feign >= 9.0.0

 Finagle >= 6.45.0

 Grizzly >= 2.1

 HTTP Kit >= 2.2.0

 HttpUrlConnection

 Jersey >= 1.17, >= 2.10

 JSF

 NanoHTTPD >= 2.2

 OkHttpd >= 3.0

 Play2 >= 2.3

 Ratpack >= 1.5

 RESTEasy >= 3.0

 Servlet >= 2.0, >= 3.0

 Spray >= 1.3

 Spring Cloud Gateway >= 2.0.2

 Spring REST >= 4.2.0

 Spring Web >= 3.2.0

 Spring Webflux >= 5.0.1

 Vaadin >= 7.0

 Vert.x-Web >= 3.3

 WebMethods Glue >= 5.0

 Wicket >= 6.0, >= 7.0

 Other:

**Java DB Clients**

Amazon DynamoDB >= 1.11

Amazon Elasticache >= 1.11

Cassandra >= 2.0, >= 3.0

Couchbase >= 2.5.5

Ehcache >= 2.0

ElasticSearch>= 1.4,>= 2.0,>= 5.0,>= 6.0

FaunaDB >= 1.2

Google Cloud BigTbl HBase 1.x >= 1.0.0

Google Cloud BigTbl HBase 2.x >= 1.1.0

Google Cloud Store >= 1.2

Hazelcast Java Client >= 3.9

JDBC >= 4

Lettuce >= 3.0, >= 4.0

MongoDB >= 2.13, >= 3.0

MongoDB Reactive >= 0.12.0

Neo4j Java Driver >= 1.5.0

Redis Jedis >= 2.8.0

Redis Lettuce >= 3.4.2, >= 4.1.2

Shade >= 1.8

SpyMemcached >= 2.10

Vert.x Redis >= 3.1.0

Other:

**Java Messaging & Async**

 ActiveMQ

 Aerospike >= 3.3, >= 4.0

 Amazon Kinesis >= 1.11

 Amazon SNS >= 1.11

 Amazon SQS >= 1.11

 Akka >= 2.3

 Camel >= 2.17

 Executor Pools

 Fork Join Pool

 gRPC >= 1.2

 HornetQ >= 2.2

 Hystrix

 IBM MQ >= 8.0

 JMS

 Kafka >= 0.8

 Mule >= 3.8

 RabbitMQ >= 3.6

 Tibco ESB

 Other:

**Java Application Server**

 IBM WebSphere

 IBM Liberty

 Oracle/BEA WebLogic

 Apache Tomcat

 Eclipse Jetty

 Glassfish / Payara

 JBoss AS / Wildfly

 Sun ONE Server

 Pega Systems AS

 Other:

**Java Other**

 Corba (Sun)

 FTP Commons >= 3.5

 FTP JSCH >= 0.1.54

 Glassfish EJB

 Google Cloud Store >= 1.2

 GWT User >= 1.2

 Java Mail

 JBoss EJB >= 5, >=7

 Lift Actor >= 2.6.3, >= 3.3.0

 LDAP

 Sun ON/RPC >= 1.1.4

 OpenEJB >= 4.7

 Quartz

 Spring Batch >= 3.0

 Tabex

 Other:

**Java Logging**

 Java Util Logging

 JBoss Log Manager >= 1.0

 Log4j >= 1.2

 Log4j 2 >= 2.4

 SLF4J >= 1.7

 Other:

## **Python Runtimes**

 CPython (>=2.7 <3.0) or >=3.3

**Python Webservers**

 uWSGI

 Gunicorn

 mod\_wsgi

 Nginx WSGI

 Other:

**Python Frameworks**

 Flask >= 0.12.0

 Django >= 1.9

 Tornado >= 4.5.3; < 6.0.0

 WSGI based frameworks

 WSGI middleware

 Other:

**Python HTTP Clients**

 aiohttp >=3.1.0

 asynqp >=0.5.1

 MySQL-python >= 1.2.5

 mysqlclient >= 1.3.14

 PyMySQL >= 0.6.7redis >= 2.10.6 < 3.0.0

 requests >= 2.16.1

 SQLAlchemy >= 1.1.5

 suds-jurko >= 0.6

 urllib3 >= 1.18.1

 Other:

## 

## **Ruby Runtimes**

**Ruby Runtime**

 MRI Ruby >= 2.1

 Other

**Ruby Messaging & Background Job Processing**

 Resque >= 1.22.0

 Sidekiq >= 2.12.0

 Other

**Ruby Memcache**

 Dalli >= 2.0.0

 Other:

**Ruby Webserver**

 Apache

 Nginx

 Passenger

 Puma

 Rhebok

 Thin

 Unicorn

 Other

**Ruby Frameworks**

 Action Controller Rails 4+5

 ActionView

 ActiveRecord >= 3.2

 Cuba >= 3.0.0

 Padrino >= 0.12.0

 Rack >= 1.4.5

 Rails API >= 0.4.0

 Rails LTS >= 2.3

 Roda >= 2.0.0

 Ruby on Rails >= 3.2

 Sinatra >= 1.1.4

 Other:

**Ruby HTTP Clients**

 Excon >= 0.21.0

 Faraday >= 0.7.6

 GRPC >= 1.0.1

 HTTP-Requestor > 1.0.0

 HTTParty via Net::HTTP

 Net::HTTP >= 2.1

 RestClient >= 1.6.0

 Other:

## **Node Runtimes**

 node.js 4.5 and above

 node.js 5.10 and above

 node.js 6.0.0 and above

 node.js 7.0.0 and above

 node.js 8.2.1 and above

 node.js 9.1.0 and above

 node.js 10.4.0 and above

 node.js 11.0.0 and above

 node.js 12.0.0 and above

 node.js 14.0.0 and above

 node.js 16.0.0 and above

 Other

**Node HTTP Clients & RPC**

 HTTP(s) servers >=1.10.0

 HTTP(s) clients >=1.10.0

 Express >=1.32.0, 1.43.0

 Fastify 1.x path templates >=1.44.0

 koa-router path templates >=1.56.0

 GRPC >=1.63.1

 GraphQL >=1.69.0

 Other

**Node DB Clients**

 Elasticsearch >=1.10.0

 MongoDB >=1.13.0

 Mongoose >=1.13.0

 MySQL >=1.29.0

 MySQL >=1.37.1

 MSSQL >=1.47.0

 Postgres >=1.44.2

 Redis >=1.31.0

 ioredis >=1.33.0

 Other:

**Node Async Technologies**

 Timers >=1.10.0

 Native Promises >=1.10.0

 Bluebird Promises >=1.35.0

 Other

**Node Messaging**

 kafka-node1 2 >=1.20.0

 RabbitMQ/amqplib1 >=1.51.0

 Other

## **.NET Runtime**

**.NET Runtime**

 .NET Framework >4.0

**.NET HTTP Clients**

 [ASP.NET](http://asp.net) Core

 [ASP.NET](http://asp.net) MVC

 [ASP.NET](http://asp.net) WebAPI

 [ASP.NET](http://asp.net) WebForms

 System.Net.Http.HttpClient

 System.Net.HttpWebRequest

 WCF

 Other:

**.NET DB Clients**

 Cassandra

 Elasticsearch (NEST)

 Memcached

 Microsoft SQL-Server (System.Data.SqlClient)

 MongoDB

 Oracle (ODP, Devart)

 PostgreSQL (Devart, NpgSql)

 Redis

 Other:

**.NET Messaging**

 Microsoft Message Queue

 RabbitMQ

 Other:

**.NET Scheduler**

 Hangfire

 Quartz

 Other:

**.NET Logging**

 NLog

 Serilog

 Log4Net

 Other:

## **POV Planning, Network and Security Requirements**

**Planning**

The Instana Observability solution consists of two general components: agents and the backend.

**Agents**

Agents are deployed into your environment(s) to discover your infrastructure, platforms (like K8s, OCP, vSphere and PCF) and applications services. They do this by leveraging [sensors](https://www.instana.com/docs/ecosystem/#agent-sensors): a sensor being that part of the agent responsible for monitoring a *specific* infra, middleware or application component/service. Agents will then provide 1 second granularity for gathered metrics as well as capturing a distributed trace for *every* application call. Note that agents can be deployed to on-prem, cloud or hybrid environments. Agents come in two flavors: dynamic and static.

Static agents can be considered traditional agents, they contain all sensors (are self-contained) and are available in Linux distros, tarball archives or MSI installers and can be installed with a 1-liner command as well.

Dynamic agents are unique in that they are “thin” (they contain no sensors). When they first start up they will discover and download only those sensors they need to monitor the various components on the host/node they are running on. They are available in Linux distros, tarball archives or MSI installers and can be installed with a 1-liner command as well. Finally, dynamic agents can update themselves *automatically* – this can be extremely beneficial in large enterprise organizations. A dynamic agent needs access to the internet to be able to reach our public repos.

Finally, the Instana agent is also available as a Docker container and can also be installed into K8s/OCP through a daemonset yaml file, helm chart or operator.

Full agent installation reference is [here](https://www.instana.com/docs/setup_and_manage/host_agent/).

**Backend**

All agents must report to a backend. Backends are available in two flavors: SaaS or on-perm.

Instana can provision a SaaS backend (literally in minutes) for the POV. Agents and users would need to be able to access it (see network requirements below). It eliminates friction starting the POV since an on-prem server or VM would not have to be procured/provisioned.

If you require an on-prem backend, we will need to size its hardware appropriately. Please consult with your Instana SA who can assist in this task. Minimum configurations begin at 16-core, 64Gb RAM w/ 500Gb of disk space (Docker is a requirement as our backend runs on Docker).

**Security Requirements**

The agent needs to run as root or privileged (sudo). Instana agents (containers) in K8s or OCP need to run privileged.

**Network Requirements**

In general, agents require one-way, outbound HTTPS network connectivity to their backend to be able to report metrics and traces (the backend *never* initiates connections to agents). A SaaS backend listens on port 443 for incoming agent connections while an on-perm backend listens on port 1444 for incoming agent connections.

Detailed network requirements for installing an on-prem backend are [here](https://www.instana.com/docs/self_hosted_instana/network-access).

Instrumented application runtimes require a network connection to port 42699 on the agent process as noted [here](https://www.instana.com/docs/setup_and_manage/host_agent/on#network-requirements).