1 CBA

Generated on: 2026-01-13 15:04:55 **Version:** 1.0

1.1 Table of Contents

1.2 Overview

CNCF CBA

The **Certified Backstage Associate (CBA)** exam demonstrates knowledge of Backstage, the open platform for building developer portals.

1.3 Exam Overview

Detail	Information
Exam Format	Multiple Choice
Number of Questions	60
Duration	90 minutes
Passing Score	75%
Certification Validity	3 years
Cost	$250 USD
Retake Policy	1 free retake

1.4 Exam Domains & Weights

Domain	Weight
Backstage Overview and Architecture	18%
Software Catalog	25%
Software Templates	20%
TechDocs	12%
Plugins	15%
Customization and Integration	10%

1.5 Key Topics

1.5.1 Software Catalog

- Entity model and kinds
- Catalog ingestion
- Entity relationships
- Ownership and metadata

1.5.2 Software Templates

- Template syntax
- Actions and steps
- Input parameters

- Scaffolding

1.5.3 Plugins

- Plugin architecture
- Frontend and backend plugins
- Plugin development basics

1.6 Study Resources

- Backstage Documentation
- CBA Curriculum
- Backstage GitHub

1.7 Navigation

- Next: Sample Questions →

1.8 Sample Practice Questions

Download PDF Version

1.9 Practice Resources

- Backstage Documentation
- Backstage Demo

1.10 Software Catalog (25%)

1.10.1 Question 1

What are the main entity kinds in Backstage Software Catalog?

Show Solution

Main entity kinds: - **Component** - A piece of software (service, website, library) - **API** - A boundary between components - **Resource** - Infrastructure (database, S3 bucket) - **System** - Collection of components and resources - **Domain** - Collection of systems - **Group** - Team or organizational unit - **User** - Individual person - **Location** - Reference to other catalog files

1.10.2 Question 2

Write a catalog-info.yaml for a backend service.

Show Solution

```yaml
apiVersion: backstage.io/v1alpha1
kind: Component
metadata:
  name: my-backend-service
  description: Backend API service
  tags:
    - java
    - spring-boot
  annotations:
    github.com/project-slug: my-org/my-backend
    backstage.io/techdocs-ref: dir:.
spec:
  type: service
  lifecycle: production
  owner: team-backend
  system: my-system
  providesApis:
    - my-api
  dependsOn:
    - resource:my-database
```

1.10.3 Question 3

How do you define relationships between entities?

Show Solution

Relationships are defined in the `spec` section:

```yaml
spec:
  # Component relationships
  owner: team-name            # Group that owns this
  system: system-name         # System this belongs to
  providesApis:               # APIs this component provides
    - api-name
  consumesApis:               # APIs this component uses
    - other-api
  dependsOn:                  # Dependencies
    - component:other-service
    - resource:database
  # System relationships
  domain: domain-name        # Domain this system belongs to
```

1.11 Software Templates (20%)

1.11.1 Question 4

Create a basic Software Template for scaffolding a new service.

Show Solution

```yaml
apiVersion: scaffolder.backstage.io/v1beta3
kind: Template
metadata:
  name: create-service
  title: Create New Service
  description: Create a new microservice
spec:
  owner: platform-team
  type: service
  parameters:
    - title: Service Details
      required:
        - name
        - owner
      properties:
        name:
          title: Service Name
          type: string
        owner:
          title: Owner
          type: string
          ui:field: OwnerPicker
        description:
          title: Description
          type: string
  steps:
    - id: fetch
      name: Fetch Template
      action: fetch:template
      input:
        url: ./skeleton
        values:
          name: ${{ parameters.name }}
          owner: ${{ parameters.owner }}
    - id: publish
      name: Publish to GitHub
      action: publish:github
      input:
        repoUrl: github.com?owner=my-org&repo=$
        {{ parameters.name }}
    - id: register
      name: Register in Catalog
      action: catalog:register
      input:
        repoContentsUrl: $
        {{ steps.publish.output.repoContentsUrl }}
        catalogInfoPath: /catalog-info.yaml
  output:
    links:
      - title: Repository
        url: ${{ steps.publish.output.remoteUrl }}
```

1.11.2 Question 5

What are common template actions?

Show Solution

Common built-in actions: - `fetch:template` - Fetch and render a template - `fetch:plain` - Fetch files without templating - `publish:github` - Create GitHub repository - `publish:gitlab` - Create GitLab repository - `catalog:register` - Register entity in catalog - `catalog:write` - Write catalog-info.yaml - `debug:log` - Log debug information - `fs:delete` - Delete files - `fs:rename` - Rename files

---

1.12 TechDocs (12%)

1.12.1 Question 6

How do you configure TechDocs for a component?

Show Solution

1. Add annotation to catalog-info.yaml:

```
metadata:
  annotations:
    backstage.io/techdocs-ref: dir:.
```

1. Create docs folder with mkdocs.yml:

```
site_name: My Service Docs
nav:
  - Home: index.md
  - API: api.md
plugins:
  - techdocs-core
```

1. Create docs/index.md with content.

1.12.2 Question 7

What are the TechDocs generation strategies?

Show Solution

Local - Generate docs on Backstage server

```yaml
techdocs:
  builder: 'local'
  generator:
    runIn: 'local'
```

External - Generate in CI/CD, store externally

```yaml
techdocs:
  builder: 'external'
  publisher:
    type: 'awsS3'
    awsS3:
      bucketName: 'my-techdocs-bucket'
```

1.13 Plugins (15%)

1.13.1 Question 8

What is the Backstage plugin architecture?

Show Solution

Backstage has three plugin types:

1. **Frontend Plugins** - React components for UI
   - Pages, cards, tabs
   - Run in browser
2. **Backend Plugins** - Node.js services
   - APIs, processors
   - Run on server
3. **Common Plugins** - Shared code
   - Types, utilities
   - Used by both frontend and backend

Plugin structure:

```
plugins/
├── my-plugin/          # Frontend plugin
│   ├── src/
│   └── package.json
├── my-plugin-backend/  # Backend plugin
│   ├── src/
│   └── package.json
└── my-plugin-common/   # Shared code
```

```
    ├── src/
    └── package.json
```

1.13.2 Question 9

How do you install a community plugin?

Show Solution

```
# Install frontend plugin
yarn add --cwd packages/app @backstage/plugin-<name>

# Install backend plugin
yarn add --cwd packages/backend @backstage/plugin-<name>-backend

# Add to app
# packages/app/src/App.tsx
import { MyPluginPage } from '@backstage/plugin-my-plugin';

// Add route
<Route path="/my-plugin" element={<MyPluginPage />} />
```

1.14 Architecture (18%)

1.14.1 Question 10

What are the main components of Backstage architecture?

Show Solution

1. **App** - Frontend React application
2. **Backend** - Node.js backend services
3. **Catalog** - Entity database and API
4. **Scaffolder** - Template execution engine
5. **TechDocs** - Documentation system
6. **Search** - Search functionality
7. **Auth** - Authentication providers

Database: PostgreSQL (production) or SQLite (development)

1.14.2 Question 11

How does catalog ingestion work?

Show Solution

Catalog ingestion process: 1. **Location** entities point to catalog files 2. **Entity Providers** fetch catalog files 3. **Processors** transform and validate entities 4. **Stitching** resolves relationships 5. **Database** stores final entities

Location types: - `url` - HTTP/HTTPS URLs - `file` - Local file paths - `github-discovery` - Auto-discover from GitHub

---

1.15 Customization (10%)

1.15.1 Question 12

How do you customize the Backstage theme?

Show Solution

```
// packages/app/src/App.tsx
import { createTheme, lightTheme } from '@backstage/theme';

const myTheme = createTheme({
  palette: {
    ...lightTheme.palette,
    primary: {
      main: '#1DB954',
    },
    navigation: {
      background: '#171717',
      indicator: '#1DB954',
    },
  },
});

// Use in app
<ThemeProvider theme={myTheme}>
  <App />
</ThemeProvider>
```

1.16 Exam Tips

1. **Know entity kinds** - Component, API, Resource, System, Domain
2. **Understand catalog-info.yaml** - Structure and annotations
3. **Practice template syntax** - Parameters, steps, actions
4. **Know TechDocs setup** - mkdocs.yml, annotations
5. **Understand plugin architecture** - Frontend, backend, common

---

← Back to CBA Overview
