Architecture

Splunk Enterprise Certified Admin

KNOWLEDGE CHEATSHEET

COLLECTION AND SEARCH

Forwarder Collect → **Indexer** Store → **Search head** Coordinate → **Results** Present

01 Platform architecture and distributed search

Components and responsibilities

- **Universal forwarder (UF)** collects and forwards data with a small footprint. It has its own installation package and built-in forwarding license; configure inputs and destinations before expecting application logs.
- **Heavy forwarder (HF)** is a Splunk Enterprise instance that can parse, mask, filter and route events before forwarding. Use it when processing requirements justify the additional resources.
- **Indexers** store searchable data and execute searches as search peers. A **search head** dispatches work, combines peer results and presents reports; it does not need to retain the indexed event corpus.
- **Distributed search** parallelizes queries across peers. Independent indexers do not automatically hold duplicate data; a failed peer can produce incomplete results without indexer replication.
- **Indexer clustering** replicates indexed data. **Search head clustering** coordinates search services and knowledge objects. These solve different availability problems.

Search peers, groups and knowledge

- **distsearch.conf** defines peers under [distributedSearch]. The **servers** setting uses a comma-separated list of peer management endpoints; the usual management port is **8089**.
- **[distributedSearch:REGION]** defines a named subset of peers. Set **default = false** when the group should require explicit selection; use **splunk_server_group=REGION** in searches.
- **Knowledge bundles** send search-time configuration and lookup knowledge from the search head to peers as needed for distributed searches. Author search-time extractions on the search tier.
- **outputlookup** writes lookup output on the search head. Distributed processing can run earlier search operations on peers before the final result is centralized.
- **splunk_server** identifies the peer serving an event. Compare counts by this field over a representative period when assessing ingestion distribution; event counts alone do not measure byte volume.

Architecture / Configuration

Splunk Enterprise Certified Admin

KNOWLEDGE CHEATSHEET

CONFIGURATION DELIVERY

App Package → **Server class** Assign → **Client** Poll → **Instance** Apply

Management roles and visibility

- **Deployment server** distributes configuration apps to deployment clients. **Deployer** distributes apps to search head cluster members. **Cluster manager** manages an indexer cluster and its configuration bundle.
- **Monitoring Console** uses deployment telemetry, including forwarded internal logs, to assess health. Enable internal-log forwarding from forwarders and search heads when central visibility is required.
- **Forwarder asset table** supports forwarder inventory and missing-forwarder status. Rebuild it when retired clients leave stale inventory; investigate missing telemetry before treating a client as healthy.
- **Search concurrency** depends on CPU-based limits and workload resources. Check CPU, memory, I/O and configured limits together; extra search heads do not repair missing data on a failed independent indexer.

02 Configuration files and deployment management

Configuration file map

- **$SPLUNK_HOME/etc** contains configuration. **inputs.conf** selects collection; **outputs.conf** selects forwarding; **indexes.conf** defines storage and retention.
- **props.conf** controls source interpretation, parsing and search-time properties. **transforms.conf** supplies referenced transformation or extraction definitions. **server.conf** holds instance-level settings.
- **authentication.conf** configures authentication strategies; **authorize.conf** defines roles. **deploymentclient.conf** configures client polling; **serverclass.conf** maps deployment clients to apps.
- **default/** contains shipped settings. Put site overrides in **local/**, such as etc/apps/<app>/local/inputs.conf, so app upgrades preserve customization. Do not edit shipped defaults.

Layering and btool

- **Global context**, highest first: etc/system/local, app local directories, app default directories, then etc/system/default. Matching stanzas merge by setting; a higher-priority value overrides the same lower-priority key.
- **Search-time context** also considers user and app scope. A user override under etc/users/<user>/<app>/local can take precedence in that context; global index-time processing does not use user-local settings.
- **Indexer cluster peer-apps** adds configuration managed by the cluster manager. Its local layer has special precedence on peers; distribute peer configuration through the cluster bundle workflow.
- **splunk btool props list --debug** shows merged on-disk values and contributing paths. Substitute inputs, outputs or another configuration name to inspect that file.
- **btool** does not prove a running process has loaded an edit. Apply the supported reload or restart, then validate runtime behavior. Use app and user context when investigating search-time differences.

Configuration / Security & Licensing

Splunk Enterprise Certified Admin

KNOWLEDGE CHEATSHEET

CONFIGURATION DELIVERY

App Package → **Server class** Assign → **Client** Poll → **Instance** Apply

Deployment apps and server classes

- **etc/deployment-apps/<app>/** is the deployment server repository. Clients install assigned apps into **etc/apps/<app>/**; placing an app in the server repository does not make it active on that server.
- **Server classes** associate client filters with apps in serverclass.conf or Forwarder Management. Filters can use client identity and machine type; an applicable exclusion overrides inclusion.
- **Deployment clients pull** assigned updates when they phone home. Reload server-side changes with **splunk reload deploy-server**; configured app settings can request a Splunk restart on clients.
- **Central app ownership** matters: updates can replace files in a managed app. Keep the authoritative configuration on the deployment server and separate unmanaged local customization from managed content.

Client polling and update flow

- **splunk set deploy-poll ds.example:8089** runs on the deployment client. It configures the deployment server target in deploymentclient.conf.
- **[target-broker:deploymentServer]** uses **targetUri** for the server address. **[deployment-client]** can set **phoneHomeIntervalInSecs = 3600** for hourly polling.
- **Restart the client** when required after manually configuring deploymentclient.conf. Confirm server reachability, client identity, class membership and assigned apps if the client fails to check in.
- **Deployment management** distributes apps and configurations; software installation and operating-system lifecycle remain separate administrative tasks. Use the cluster-specific distribution role for clustered members.

03 Users, authentication and licensing

Users, roles and knowledge permissions

- **Native users** need a username, password and appropriate role assignments. Create users through Splunk Web, CLI or REST API; operating-system accounts do not automatically become Splunk users.
- **etc/passwd** stores native credential records with password hashes. **authentication.conf** selects authentication behavior; it is not the native password database.
- **Roles** grant capabilities, searchable indexes, search filters and job limits. **user** supports ordinary searches; **power** adds broader shared-object abilities; **admin** manages the platform and roles.

Security & Licensing

Splunk Enterprise Certified Admin

KNOWLEDGE CHEATSHEET

ACCESS AND ENTITLEMENT

- **Authentication** Identity
- **Roles** Permissions
- **MFA** Verification
- **License** Entitlement

- **Role inheritance** combines permissions from imported roles. Access can broaden when several roles are assigned; default selecting **srchFilter** expressions combine with OR, while eliminating filters use AND.
- **Object sharing and permissions** control private, app and global knowledge access. Use role and index permissions together; separate search heads can isolate team knowledge administration but still need data access controls.

LDAP, SAML and scripted authentication

- **LDAP** validates directory credentials and maps directory groups to Splunk roles. Group names need not equal role names; the mapping provides authorization.
- **LDAP strategies** are checked in configured order. Native accounts take precedence for the same username; manage directory passwords and group membership in the directory.
- **SAML** delegates authentication to an identity provider and maps asserted groups to roles. MFA for this flow belongs at the identity provider.
- **Scripted authentication** integrates custom external systems such as RADIUS or PAM through the authentication API. Protect integration credentials and test authentication and role lookup together.

Duo multifactor authentication

- **Splunk-managed Duo MFA** adds a secondary challenge after native, LDAP or scripted primary authentication. SAML and proxy SSO follow their external authentication flow.
- **Duo configuration** uses provider-issued integration credentials and an API hostname. Obtain them from the Duo administrator and protect all secret values.
- **Primary authentication → secondary challenge → session** is the core login sequence. Choose explicitly whether provider unavailability permits login or denies it; test the selected outage behavior.

License volume and administration

- **Ingest-based licensing** meters eligible incoming data volume. Historical backfill consumes ingestion allowance too; schedule it against daily headroom alongside normal live data.
- **License manager, stacks and pools** organize entitlements and allocations to license peers. Pools apply to indexer peers rather than individual indexes; **strict_pool_quota** affects pool enforcement.
- **License files** reside under etc/licenses. Enterprise, trial, free and forwarder licenses enable different features; expiration and enforcement depend on the installed license and software.

Security & Licensing / Data Inputs

Splunk Enterprise Certified Admin

KNOWLEDGE CHEATSHEET

INPUT CHOICES

- **Monitor** — Files
- **Script** — Program output
- **Network** — TCP or UDP
- **HEC** — Token and HTTP

- **Metered volume is not disk footprint.** Compression, replicated buckets and deletion of indexed data do not reverse already metered ingestion. Splunk internal logs and qualifying summary data have different accounting from ordinary source logs.
- **Event and metrics ingestion** can use different accounting rules. Review license usage telemetry before a large import; adding an appropriate entitlement is a license-manager operation.

04 Data inputs and HTTP Event Collector

File and directory monitoring

- **[monitor:///var/log/app.log]** selects a local absolute path in inputs.conf. Set **disabled = 0**, **index** and **sourcetype** deliberately, and verify the service account can read the file.
- **Directory monitors** discover new files and normally recurse through subdirectories. ***** matches within a path segment; **...** can span directory levels. Path wildcards differ from regex filters.
- **whitelist** and **blacklist** filter monitor paths with regular expressions. A matching blacklist excludes a file even if it matches the whitelist.
- **ignoreOlderThan** filters by file modification age, not timestamps of individual events. **followTail** can skip pre-existing content when first discovering a file; it is not an event-age filter.
- **splunk add oneshot /path/file.log -index incident** ingests a static file once. A continuous monitor tracks future changes; uploading a sample is useful for testing without creating a persistent monitor.

Scripts and Windows collection

- **Scripted inputs** run a program and ingest standard output. Package scripts in etc/apps/<app>/bin; other supported script locations include etc/system/bin and bin/scripts.
- **[script://...]** defines the executable and **interval** controls scheduling. **interval = 0** restarts a script when it exits; **interval = -1** runs it once per Splunk startup.
- **Windows inputs** can collect event logs and performance counters locally. Supported remote **WMI** collection uses **wmi.conf**, Windows permissions and remote connectivity.
- **MonitorNoHandle** supports particular Windows file-access situations where ordinary monitoring is obstructed by file locks. Match the input type to the source and its supported platform.

Data Inputs / Event Processing

Splunk Enterprise Certified Admin

KNOWLEDGE CHEATSHEET

INPUT CHOICES

- **Monitor** — Files
- **Script** — Program output
- **Network** — TCP or UDP
- **HEC** — Token and HTTP

Network listeners and buffering

- **TCP or UDP input** requires a protocol and listening port, defined in inputs.conf. Open the corresponding firewall path and avoid confusing a raw network input with a Splunk-to-Splunk receiver.
- **connection_host** can derive host metadata from sender IP or DNS, or leave it unchanged with none. Sender restrictions and explicit host, source, sourcetype and index settings refine collection.
- **queueSize** controls memory buffering. **persistentQueueSize** adds disk buffering for supported input types, improving survival across interruptions; it cannot recover traffic never received.
- **UDP** provides no transport acknowledgement or backpressure. TCP and durable upstream buffering improve delivery reliability, but the complete sender-to-index path still needs validation.

HEC requests and acknowledgement

- **HTTP Event Collector (HEC)** accepts events using a token without requiring a forwarder. Send **Authorization: Splunk <token>** to the enabled HTTPS collector endpoint, commonly on port **8088**.
- **/services/collector/event** accepts JSON envelopes with an **event** member and optional metadata. A curl request uses **-H "Authorization: Splunk <token>"** and **-d '{"event":"ready"}'**. Batch concatenated envelopes; **/services/collector/raw** accepts raw text.
- **Indexer acknowledgement** is enabled for the HEC token with **useACK = true**. HTTP acceptance alone does not confirm that processing and indexing completed.
- **X-Splunk-Request-Channel** carries a client-generated **GUID**. Retain returned **ackId** values and poll **/services/collector/ack** using the same channel to confirm indexing.
- **Retry logic** must handle HTTP errors, pending acknowledgements and possible duplicates. Keep HEC token permissions, destination indexes, TLS trust and client buffering consistent.

05 Forwarding and event processing

Forwarding destinations and load balancing

- **outputs.conf** configures **[tcpout] defaultGroup** and **[tcpout:<group>] server**. Multiple comma-separated receivers within one group enable load balancing; multiple selected groups can clone data.
- **splunk add forward-server indexer.example:9997** adds a destination. **splunk list forward-server** reports forwarding targets. The indexer must separately enable a Splunk receiving port.
- **autoLBFrequency** controls time-based destination switching. Verify receiver ports, DNS results and network access when distribution is uneven; continuous data can delay a safe switch.

Event Processing

Splunk Enterprise Certified Admin

KNOWLEDGE CHEATSHEET

INGESTION PIPELINE

Input Collect → **Parsing** Form events → **Transforms** Refine → **Indexing** Persist

- **EVENT_BREAKER_ENABLE** and **EVENT_BREAKER** let a UF identify safe event boundaries for forwarding. They differ from the parser's **LINE_BREAKER** settings.
- **_TCP_ROUTING** in an input selects named tcpout groups. **_SYSLOG_ROUTING** refers to syslog output routing; it is a different output mechanism.

Forwarder reliability and transport security

- **useACK = true** retains sent blocks until the receiver acknowledges them. Its wait queue is three times a size-based **maxQueueSize**; a 7 MB output queue therefore uses a 21 MB wait queue.
- **Backpressure** propagates when output queues fill. With cloned destinations, a blocked target group can impede other groups; assess queue and dropping policies against delivery requirements.
- **compressed = true** controls forwarding compression in outputs.conf. **TLS** encrypts transport and needs compatible receiver configuration plus certificate validation; an IP address alone says nothing about encryption.
- **Cross-platform forwarding** works between supported forwarder and receiver operating systems. Confirm software compatibility and connectivity rather than requiring matching operating systems.

Pipeline phases and configuration placement

- **Input** collects streams, attaches initial metadata and handles character encoding. **Parsing** establishes events, timestamps and transformations. **Indexing** writes searchable structures and raw data.
- **CHARSET** in props.conf selects source encoding; Splunk uses UTF-8 internally. Apply encoding configuration where input processing occurs.
- **First parsing instance** handles ordinary event breaking and raw-data transforms: the HF if present, otherwise the indexer. Parsed data usually bypasses those parsing rules downstream.
- **INDEXED_EXTRACTIONS** handles supported structured data such as CSV or JSON and is a special processing path that can run on a UF. Avoid duplicate field extraction downstream.
- **Search time** applies knowledge such as field extractions and aliases. Search-time masking changes presentation; protect stored sensitive data by applying required redaction before indexing.

Event boundaries and timestamps

- **props.conf stanzas** use [<sourcetype>], [source::<path>] or [host::<name>]. When matching stanza types conflict, source properties outrank host properties, which outrank sourcetype properties.
- **SHOULD_LINEMERGE = false** with a suitable **LINE_BREAKER** is efficient for predictable boundaries. The usual newline expression is **([\r\n]+)**; its first capture identifies the separator to discard.

Event Processing / Index Management

Splunk Enterprise Certified Admin

KNOWLEDGE CHEATSHEET

INGESTION PIPELINE

Input Collect → **Parsing** Form events → **Transforms** Refine → **Indexing** Persist

- **Multiline and XML events** need boundaries matching complete records. A custom LINE_BREAKER can retain multiline records with line merging disabled; enabling line merging is a separate assembly approach.
- **TIME_PREFIX** locates the timestamp, **TIME_FORMAT** describes its syntax and **MAX_TIMESTAMP_LOOKAHEAD** bounds the characters examined after the prefix. Include fractional seconds and offsets when present.
- **TZ** supplies a timezone when needed; explicit event offsets take priority. Missing date information requires timestamp inference. **TRUNCATE** limits event length; setting it to 0 removes that length limit.

Masking, filtering and routing

- **SEDCMD-<name>** in props.conf performs sed-style substitutions on raw events. Capture groups preserve selected text; validate that the expression neither leaks secrets nor removes needed context.
- **TRANSFORMS-<class>** in props.conf references transforms.conf stanzas. **REGEX** matches data, **DEST_KEY** selects the changed target and **FORMAT** supplies the replacement.
- **DEST_KEY = _raw** changes raw text. **DEST_KEY = queue** with **FORMAT = nullQueue** drops matching events; **DEST_KEY = _MetaData:Index** can assign an index.
- **Search-time extractions** use mechanisms such as **EXTRACT**, **REPORT** and **FIELDALIAS**. These differ from index-time TRANSFORMS rules and can be scoped through app or user context.
- **Data Preview** validates event boundaries, timestamps and source interpretation on representative samples. Test routing and other pipeline-specific behavior in the actual processing path before rollout.

06 Indexes, retention and ingestion troubleshooting

Buckets and storage paths

- **Indexes** contain buckets of events. Separate indexes support distinct retention policies, access controls and operational data organization.
- **Hot → warm → cold → frozen** describes the normal lifecycle. Hot buckets accept writes; warm and cold buckets remain searchable. Frozen data is deleted unless an archive process is configured.
- **Thawed** buckets contain restored archived data and become searchable again. Thawed storage is separately managed; frozen is a lifecycle state rather than a required searchable directory.
- **indexes.conf** uses **homePath** for hot and warm buckets, **coldPath** for cold buckets and **thawedPath** for restored buckets. Typical directories are db, colddb and thaweddb.
- **Storage tiers** can place hot and warm data on faster storage and cold data on a supported slower tier. Confirm storage support and performance for the indexing workload.

Index Management

Splunk Enterprise Certified Admin

KNOWLEDGE CHEATSHEET

BUCKET LIFECYCLE

Hot Writable → **Warm** Searchable → **Cold** Searchable → **Frozen** Archive or delete

Rollover, retention and integrity

- **maxDataSize** helps determine hot bucket size. **maxWarmDBCount** limits warm bucket count and can trigger warm-to-cold movement; hot roll conditions also include age and other limits.
- **frozenTimePeriodInSecs** controls time-based freezing. A bucket qualifies after its newest event is old enough; retention operates on whole buckets rather than deleting each expired event immediately.
- **maxTotalDataSizeMB** can freeze old buckets when an index reaches its size limit. Time and size policies interact; a nominal retention duration does not guarantee that many days will fit.
- **coldToFrozenDir** or **coldToFrozenScript** supports archiving at freezing. Without an archive mechanism, frozen data is removed from the index.
- **enableDataIntegrityControl** in indexes.conf enables raw-data integrity checking. **splunk check-integrity -index <name>** checks supported local indexed data for changes; it does not validate application configuration.

Checkpoints and missing-data triage

- **Fishbucket** tracks monitored-file fingerprints and read positions on the collecting instance. Restarting Splunk normally resumes collection; changing an index assignment does not reread old content.
- **Targeted btprobe reset** can clear a file checkpoint for deliberate reingestion. Operate on the collecting UF or instance, preserve existing state and plan for duplicates and license volume.
- **_internal** contains Splunk operational logs; **_audit** records audit activity; **_introspection** records resource telemetry. These support health diagnosis independently of ordinary application indexes.
- **No incoming data:** check app-local inputs.conf, disabled state, path, permissions, filename filters and checkpoints; then inspect outputs, receiver reachability, parsing queues and destination index existence.
- **Search scope matters:** verify index access, sourcetype, source, host and time range. Bad timestamp parsing can make successfully indexed data appear missing; inspect internal errors and ingestion telemetry too.